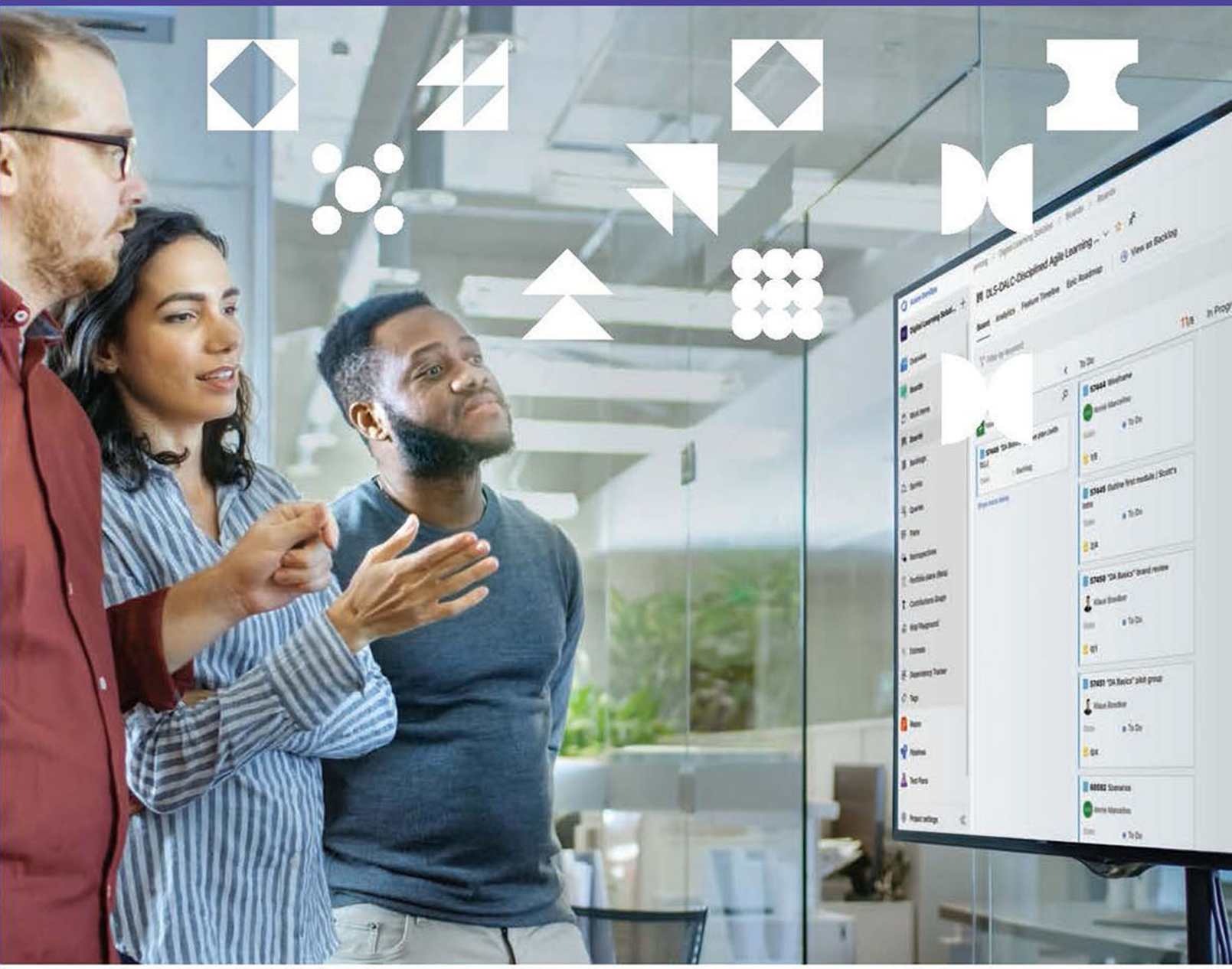
Introduction to Disciplined Agile Delivery—Second Edition

A Small Agile Team's Journey From Scrum to Disciplined DevOps



Disciplined Agile

Mark Lines and Scott W. Ambler

VISIT…

Introduction to Disciplined Agile Delivery

Second Edition

A Small Agile Team's Journey From Scrum to Disciplined DevOps

Mark Lines and Scott W. Ambler

v1.1

Library of Congress Cataloging-in-Publication Data

Names: Lines, Mark, 1964-author. | Ambler, Scott W., 1966-author.
Title: Introduction to disciplined agile delivery: a small agile team's journey from Scrum to Disciplined DevOps / Mark Lines and Scott W. Ambler. Other titles: Disciplined agile delivery. Description: Second Edition. | Newtown Square : Project Management Institute, Inc., 2020. | Revised edition of the authors' Disciplined agile delivery, 2012. | Includes bibliographical references and index. | Summary: "This guide to Disciplined Agile Delivery (DAD) provides the foundation from which to scale agile and lean software development strategies, enabling teams to succeed in the unique situations they face-remembering that context counts. Starting with a Scrum-based approach, we'll show you how the team learns from their experiences and evolves into a lean life cycle, then to a continuous delivery life cycle within a DevOps environment. We begin with an overview of DAD and work through a case study that describes a team's learnings through several releases of a mission-critical solution"-- Provided by publisher.
Identifiers: LCCN 2020022051 | ISBN 9781628256543 (paperback) | ISBN 9781628256550 (epub) | ISBN 9781628256567 (kindle edition) | ISBN 9781628256574 (pdf)
Subjects: LCSH: Agile software development.
Classification: LCC QA76.76.D47 A4253 2020 | DDC 005.1/112--dc23
LC record available at https://lccn.loc.gov/2020022051

ISBN: 978-1-62825-654-3

Published by: Project Management Institute, Inc.
14 Campus Boulevard
Newtown Square, Pennsylvania 19073-3299 USA
Phone: +1 610 356 4600
Fax: +1 610 356 4647
Email: customercare@pmi.org
Internet: www.PMI.org

©2020 Project Management Institute, Inc. All rights reserved.

Our copyright content is protected by U.S. intellectual property law that is recognized by most countries. To republish or reproduce our content, you must obtain our permission. Please go to http://www.pmi.org/permissions for details.

PMI, the PMI logo, PMBOK, OPM3, PMP, CAPM, PgMP, PfMP, PMI-RMP, PMI-SP, PMI-ACP, PMI-PBA, PROJECT MANAGEMENT JOURNAL, PM NETWORK, PMI TODAY, PULSE OF THE PROFESSION and the slogan MAKING PROJECT MANAGEMENT INDISPENSABLE FOR BUSINESS RESULTS are all marks of Project Management Institute, Inc. **For a comprehensive list of PMI trademarks, contact the PMI Legal Department.** All other trademarks, service marks, trade names, trade dress, product names and logos appearing herein are the property of their respective owners. Any rights not expressly granted herein are reserved.

To place a Trade Order or for pricing information, please contact Independent Publishers Group:

Independent Publishers Group
Order Department
814 North Franklin Street
Chicago, IL 60610 USA
Phone: 800 888 4741
Fax: +1 312 337 5985
Email: orders@ipgbook.com (For orders only)

For all other inquiries, please contact the PMI Book Service Center.

PMI Book Service Center
P.O. Box 932683, Atlanta, GA 31193-2683 USA
Phone: 866 276 4764 (within the U.S. or Canada) or +1 770 280 4129 (globally)
Fax: +1 770 280 4113
Email: info@bookorders.pmi.org

Printed in the United States of America. No part of this work may be reproduced or transmitted in any form or by any means, electronic, manual, photocopying, recording, or by any information storage and retrieval system, without prior written permission of the publisher.

The paper used in this book complies with the Permanent Paper Standard issued by the National Information Standards Organization (Z39.48—1984).

10 9 8 7 6 5 4 3 2 1

Contents

Chapter 1 Introduction 1

Chapter 2 Reality Over Rhetoric 3

Chapter 3 Disciplined Agile Delivery in a Nutshell 5

Chapter 4 Introduction to the Case Study 25

Chapter 5 Inception 29

Chapter 6 Construction Iteration C1 39

Chapter 7 Construction Iteration C2 47

Chapter 8 Construction Iteration C3 53

Chapter 9 Construction Iteration C7 57

Chapter 10 Construction Iteration C10 61

Chapter 11 Transition 65

Chapter 12 The Road to Disciplined DevOps 67

Chapter 13 Closing Thoughts 75

Appendix: The Disciplined Agile Tool Kit 77

References 81

Index 83

About the Authors 87

Acknowledgments

We'd like to thank Beverley Ambler, Rod Bray, David Dame, Matt Everard, Louise Lines, Glen Little, Valentin-Tudor Mocanu, Kristen Morton, David Shapiro, Paul Sims, and Michael Vizdos for their feedback and input into the development of this book. We couldn't have done it without you.

Chapter 1

Introduction

Many organizations are struggling to be successful with mainstream agile methods such as Scrum. Sometimes, the impulse is to give up and try the *next great thing* such as Lean or Scaled Agile Framework (SAFe®). The reality is that the source of failure of existing agile adoptions can often be traced to either the misapplication of core agile principles or a naïve approach to scaling agile and the need to address enterprise concerns.

Adding to the confusion is the constant bickering in the agile community about which method is best between Scrum, Extreme Programming (XP), Kanban, SAFe, and others. The reality is that most organizations can benefit from a number of these strategies, albeit with some consistency that a common tool kit can provide. This is where the Disciplined Agile™ (DA) tool kit comes in.

DA is a hybrid of existing methods that provides the flexibility to use various approaches, as well as plugging some gaps not addressed by mainstream agile methods. In a nutshell, DA, and more importantly, the Disciplined Agile Delivery (DAD) portion of it that is the focus of this book, is "pragmatic agile." It describes proven strategies to adapt and scale agile initiatives that suit the unique realities of your enterprise, without having to figure it all out by yourself.

The book *Choose Your WoW! A Disciplined Agile Delivery Handbook for Optimizing Your Way of Working* is the definitive guide for DAD. We have key information about DA, in general, at PMI.org/disciplined-agile.

While *Choose Your Wow* is a great overview and reference guide, it is over 400 pages. We felt that it would be useful to summarize DAD into a quick read to show how it can be applied in a typical situation. That thinking led to the first edition of *Introduction to Disciplined Agile Delivery*, published in the third quarter of 2015. We have found that organizations that simply take a bit of time to understand what DAD is, as well as what it is not, see very quickly the obvious benefits of DAD.

We wrote this short book in the hope that after having invested just an hour or two to learn about what DAD actually is, you are convinced that it is worthwhile to further investigate the possible fit for DAD within your organization.

Some quick facts about DAD:

- DAD is the delivery portion of the Disciplined Agile (DA) tool kit. DAD is a key component of Disciplined DevOps, which in turn is part of DA's value stream layer, which is part of the Disciplined Agile Enterprise (DAE). The book *An Executive's Guide to the Disciplined Agile Tool Kit* provides an overview of DA, as does the appendix at the back of the book.

- DA is resonating within organizations around the world. Organizations that have adopted, or are in the process of adopting, DA include large financial institutions, software companies, e-commerce companies, restaurant chains, government agencies, and many others. By adoption, we mean either planning or actively implementing it across their entire organization, not just with one or two teams.
- Although DAD was originally developed at IBM, it is now the intellectual property of Project Management Institute (PMI), and is available at no cost to PMI members.
- The certification program is described at PMI.org.
- DA is not a replacement for existing agile and lean methods. It complements them and, in many cases, extends them to be enterprise ready.

Chapter 2

Reality Over Rhetoric

One of the reasons why DAD is quickly growing in popularity is that it acknowledges how organizations that are effectively scaling agile are *really* doing things. We think that it is important to clear up some of the misconceptions regarding the hype of agile purism versus what we have found to be really happening based on both our hands-on experience with many clients around the world and our comprehensive industry research.[1]

Myth	Reality
Agile teams don't do requirements or planning.	The average agile team spends about one month doing some up-front planning and requirements modeling. While DAD seeks to minimize this work, we acknowledge this reality and suggest that teams new to agile spend a few weeks in an Inception phase to complete the work in a minimal, yet sufficient, fashion.
DAD is a form of "WaterScrumFall," with lots of up-front planning and requirements (Water), and testing and deployment at the end (Fall), with Scrum in the middle.	While DAD acknowledges that some prework is required to secure funding, among other precoding activities, DAD suggests you minimize the work in Inception. Similarly, while DAD recognizes that the common pattern in enterprises is to deploy solutions in a structured fashion, which we describe as the Transition phase, the disciplined testing practices in the Construction phase, such as continuous integration, should minimize the transition effort and end-of-life-cycle testing. For more advanced applications of DAD, the teams may not even require an explicit Inception or Transition phase, as described in DAD's Continuous Delivery: Agile and Continuous Delivery: Lean life cycles.
"Governance" is an agile dirty word. The agile concept of self-organization means that enterprises should not interfere with how agile teams deliver their software.	Governance is actually a good thing when it's done in an agile/lean manner. Sponsors, and the enterprise as a whole, deserve to know that their investments are being properly spent and that the risk of delivery is monitored and controlled, albeit in a lightweight, agile fashion. DAD provides specific guidance to fulfill this responsibility in a relatively painless fashion.
DAD is complicated and would be disruptive to adopt.	DAD is quite simple to adopt, especially if your organization is familiar with common agile practices. The good news is that DAD provides guidance that addresses why some existing agile implementations are struggling, and DAD can help to bring these implementations back on track.
SAFe is *the* solution to scaling agile.	For large organizations that have delivery teams greater than 100 people, SAFe may indeed be a good fit for *some* teams. However, most organizations have a mix of small- to medium-sized teams delivering solutions for multiple lines of business. In these situations, SAFe may not be suitable. While SAFe is suitable in a specific context, DAD is much more flexible and adaptable to a broader range of situations in the enterprise.

[1] See Ambysoft.com/surveys/.

Chapter 3

Disciplined Agile Delivery in a Nutshell

Key Points

- DAD is the delivery portion of the Disciplined Agile (DA) tool kit, not just another methodology.
- If you are using Scrum, XP, or Kanban, you are already using variations of a subset of DAD.
- DAD provides six life cycles to choose from; it doesn't prescribe a single way of working. Choice is good.
- DAD focuses on achieving common goals in an agile manner, not the production of specific work products or following a prescriptive agile strategy.
- DAD addresses key enterprise concerns not described by mainstream methods such as Scrum.
- DAD is complementary to SAFe, yet far less prescriptive and more practical for most enterprises.
- DAD shows how agile works from beginning to end.
- DAD provides a flexible foundation from which to scale mainstream methods.
- While DAD's philosophy is consistent with that of the *Manifesto for Agile Software Development* (Agile Manifesto), it includes additional guidance to be effective in more complex enterprise situations.
- It is *not* difficult to get started with DAD.

Many organizations start their agile journey by adopting Scrum because it describes a good strategy for leading agile software teams. However, Scrum is only a small part of what is required to deliver sophisticated solutions to your stakeholders. Invariably, teams need to look to other methods to fill in the process gaps. When looking at other methods, there is considerable overlap and conflicting terminology that can be confusing to practitioners, as well as outside stakeholders. Worse yet, people don't always know where to look for advice, or even know what issues they need to consider.

To address these challenges, Disciplined Agile Delivery (DAD) is a people-first, learning-oriented, hybrid agile approach to IT solution delivery. Delivery (DAD) provides a more cohesive approach to agile solution delivery. It supports several risk value delivery life cycles, is goal driven, enterprise aware, scalable, and reflects the realities of enterprise solution delivery. DAD can—and should—be used to extend Scrum to address the critical aspects of software development that Scrum purposefully leaves up to you.

There are clearly some interesting aspects to DAD:

1. **Hybrid.** DAD is an agnostic, hybrid approach that puts proven strategies from Scrum, Agile Modeling (AM), Extreme Programming (XP), Unified Process (UP), Kanban, Lean Software Development, SAFe, and several other methods into context.
2. **Full-delivery life cycle.** DAD extends the construction-focused life cycle of Scrum to address the full beginning-to-end delivery life cycle, from team initiation all the way to delivering the solution to its end users.

3. **Support for multiple life cycles.** DAD also supports lean, continuous delivery, and exploratory versions of the life cycle. Unlike other agile methods, DAD doesn't prescribe a single life cycle because it recognizes that one process size does not fit all.
4. **Complete.** DAD includes advice about how development, modeling, documentation, and governance strategies fit together in a streamlined whole. We like to say that DAD does the "process heavy lifting" that other methods leave up to you.
5. **Context sensitive.** Instead of the prescriptive approach seen in other agile methods, including Scrum, DAD promotes what we call a goal-driven or objective-driven approach. In doing so, DAD provides contextual advice regarding viable alternatives and their trade-offs, enabling you to tailor DAD to effectively address the situation in which you find yourself. By describing what works, what doesn't work, and more importantly, why, DAD helps you to increase your chance of adopting strategies that will work for you.

People First: Roles in Disciplined Agile Delivery

As shown in Figure 3.1, DAD suggests a robust set of roles for agile solution delivery. A common question that we get is what is the difference between primary and supporting roles? Primary roles exist in all DAD teams regardless of scale. Supporting roles, however, typically occur only at scale and sometimes only for a temporary period of time. Another common question that we get is: "Why are there so many roles?" Scrum has three roles—scrum master, product owner, and team member—so why does DAD need 10? The key issue is one of scope. Scrum mostly focuses on leadership and change management aspects during Construction, and therefore has roles that reflect this. DAD, on the other hand, explicitly focuses on the entire delivery life cycle and all aspects of solution delivery, including the technical aspects that Scrum leaves out. So, with a larger scope comes more roles. For example, DAD encompasses agile architecture issues so it includes an architecture owner role. Scrum doesn't address architecture, so it doesn't include this role.

An Agnostic, Hybrid Tool Kit

Disciplined Agile (DA) is an agnostic, hybrid tool kit that builds upon the solid foundation of other methods and software process frameworks. The DAD portion of DA adopts practices and strategies from existing sources and provides agnostic advice for when and how to apply them together. In one sense, methods such as Scrum, Extreme Programming (XP), Kanban, and Agile Modeling (AM) provide the process bricks, and DAD provides the mortar to fit the bricks together effectively, as depicted in Figure 3.2.

One of the great advantages of agile and lean software development is the wealth of practices, techniques, and strategies available to you. This is also one of its greatest challenges, because without something like the DAD, it's difficult to know which practices to choose and how to fit them together. Worse yet, many teams new to agile will treat a method like Scrum or SAFe as if it's a recipe, ignoring advice from other sources and thereby getting into trouble.

Choice Is Good: Full-Delivery Life Cycles

The focus of DAD is on delivery, although remember that other aspects of IT and your organization still exist, such as data management, enterprise architecture, operations, portfolio management, and more. A full-system/product life cycle goes from the initial concept for the product, through

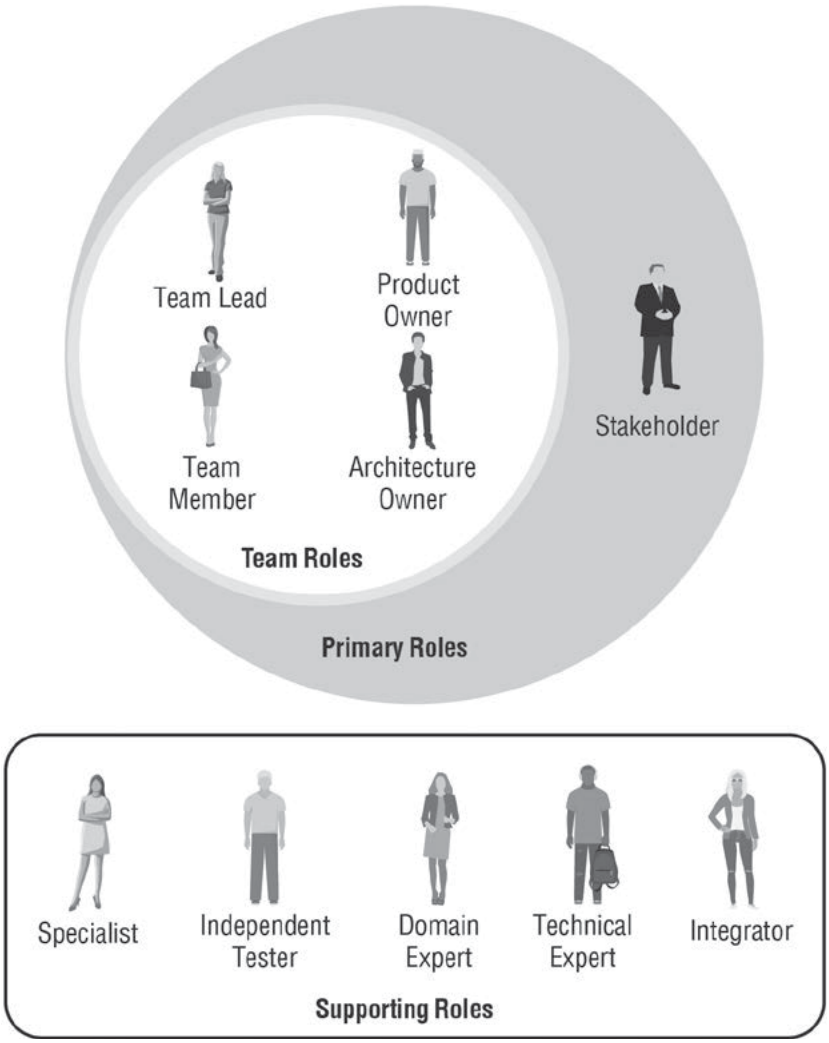


Figure 3.1 The roles of Disciplined Agile Delivery.

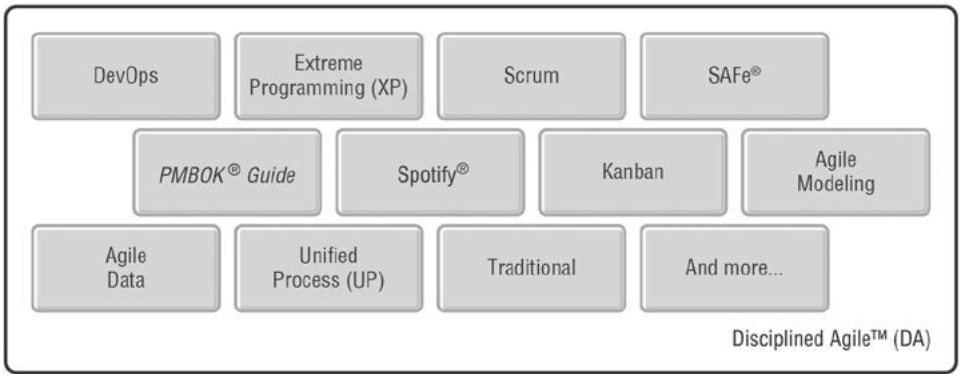


Figure 3.2 DAD combines strategies from a range of sources.

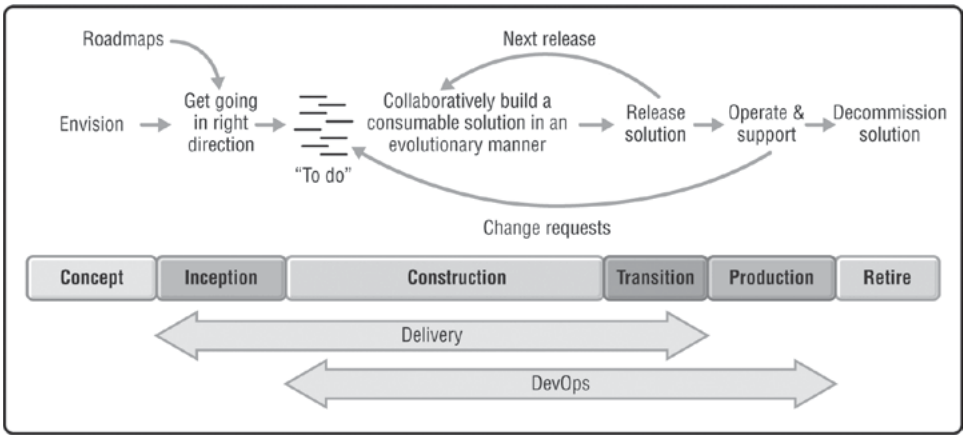


Figure 3.3 The delivery life cycle as a subset of the system life cycle.

delivery, to operations and support, and often includes many iterations of the delivery life cycle. Figure 3.3 depicts a high-level view of the DAD life cycle. The inner three phases—Inception, Construction, and Transition—form the delivery portion of the life cycle. During delivery, you incrementally build an increasingly more consumable solution.

Obviously, there's more to DAD than what this high-level diagram shows. DAD, because it's not prescriptive and strives to reflect reality as best it can, supports several versions of a delivery life cycle. Six versions of the life cycle follow: the Agile life cycle that extends the Scrum Construction life cycle with proven ideas from Unified Process to support early mitigation of risk and lightweight governance; the Lean life cycle based on Kanban; the Continuous Delivery: Agile life cycle; the Continuous Delivery: Lean life cycle; the Exploratory life cycle based upon a Lean Startup approach; and the Program life cycle for a team of teams. DAD teams will adopt the life cycle that is most appropriate for their situation and then tailor it appropriately. The visual depictions of the life cycles are shown first, then descriptions follow.

Explicit Phases Make Agile More Palatable to Management

As shown in Figure 3.4, DAD life cycles can have phases. Daniel Gagnon has been at the forefront of agile practice and delivery for almost a decade in two of Canada's largest financial institutions. He had this to say about using DA as an overarching framework: "At both large financials that I have worked in, I set out to demonstrate the pragmatic advantages of using DA as a 'top-of-the-house' approach. Process tailoring in large, complex organizations clearly reveals the need for a large number of context-specific implementations of the four (now six) life cycles, and DA allows for a spectrum of possibilities that no other framework accommodates. However, I call this 'structured freedom,' as all choices are still governed by DA's application of Inception, Construction, and Transition with lightweight, risk-based milestones. These phases are familiar to project management offices (PMOs), which means that we aren't carrying out a frontal assault on their fortified position, but rather introducing governance change in a lean, iterative, and incremental fashion."

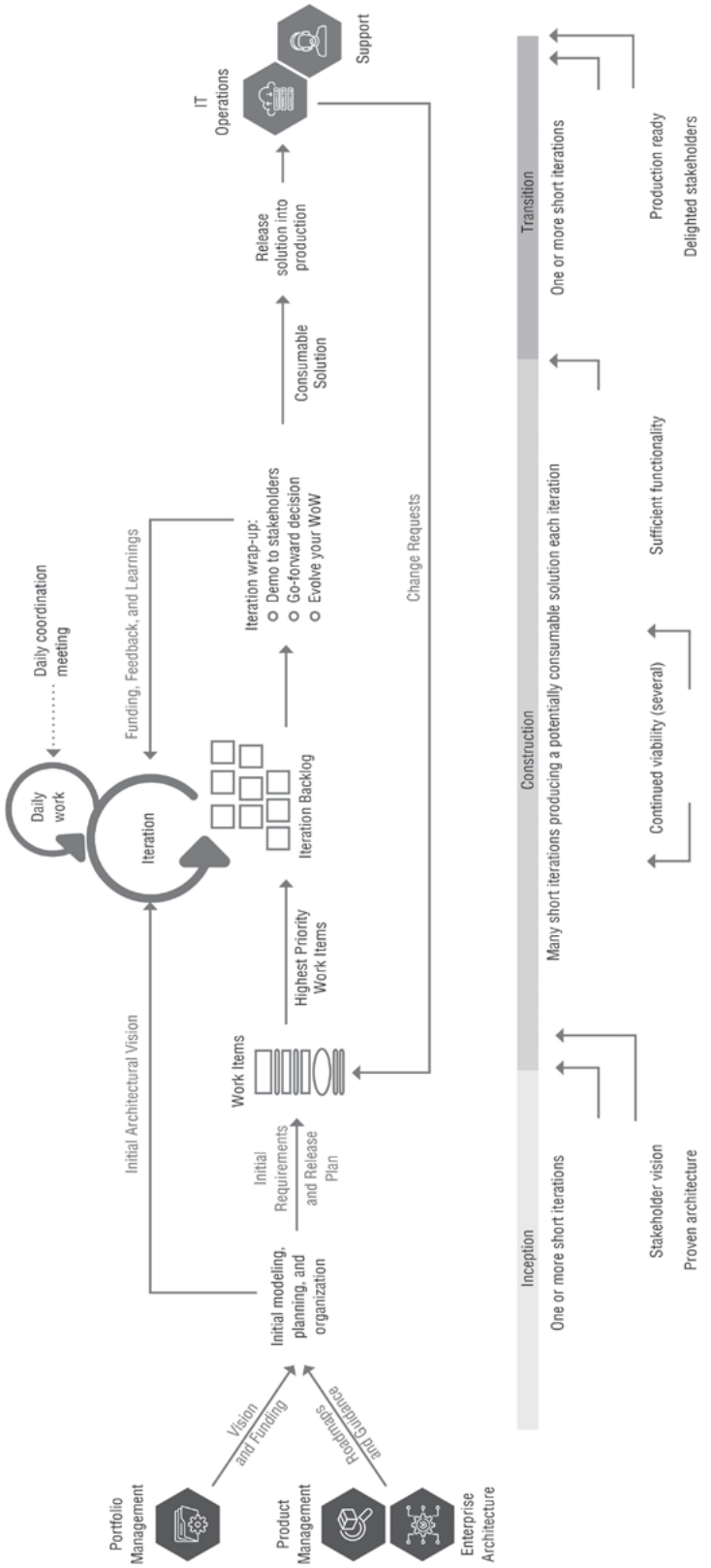


Figure 3.4 DAD's Agile life cycle.

Consumable Solutions Over Working Software

The Agile Manifesto suggests that we measure progress based upon "working software." But what good is that when the customer can't—or worse—doesn't want to use it? What we deliver should be consumable, which is *functional + usable + desirable*. Additionally, what we produce usually is not just software. There may be business changes and other supporting deliverables, so we suggest striving to deliver "consumable solutions."

In general, we suggest the following guidance regarding which life cycles fit in certain circumstances:

Agile life cycle. This life cycle, shown in Figure 3.4, is based largely upon Scrum and XP with a set of timeboxed iterations (sprints) being the core of the Construction phase. It is the most commonly used life cycle that is suitable in these types of situations:

- The work is primarily enhancements or new features.
- The work can be identified, prioritized, and estimated early in the process.
- The team is new to agile.
- The team is familiar with Scrum and XP.
- The team is typically working on a project.

Note that while the diagram shows that you should have increments of a consumable solution at the end of each iteration, for a new product or solution you may not have something truly consumable until after having completed several iterations.

Continuous Delivery: Agile life cycle. This life cycle, shown in Figure 3.5, is a natural progression from the Agile life cycle. Teams typically evolve to this life cycle from the Agile life cycle, often adopting iteration lengths of one week or less. The key difference between this and the Agile life cycle is that the continuous delivery life cycle results in a release of new functionality at the end of each iteration, rather than after a set of iterations. Teams require a mature set of practices around continuous integration and continuous deployment and other DevOps strategies. This life cycle is suitable when:

- Solutions can't be delivered to stakeholders on a frequent and incremental basis.
- Work remains relatively stable within an iteration.
- Organizations have streamlined deployment practices and procedures.
- There is a critical need to get value into the hands of stakeholders rapidly, before the entire solution is complete.
- Teams have mature DevOps practices in place, including continuous integration, continuous deployment, and automated regression testing.
- The team is long-lived (stable), and is working on a series of releases over time.

Lean life cycle. This life cycle, shown in Figure 3.6, promotes lean principles such as minimizing work in process, maximizing flow, having a continuous stream of work (instead of fixed iterations), and reducing bottlenecks. New work is pulled from the work item pool as the team has capacity.

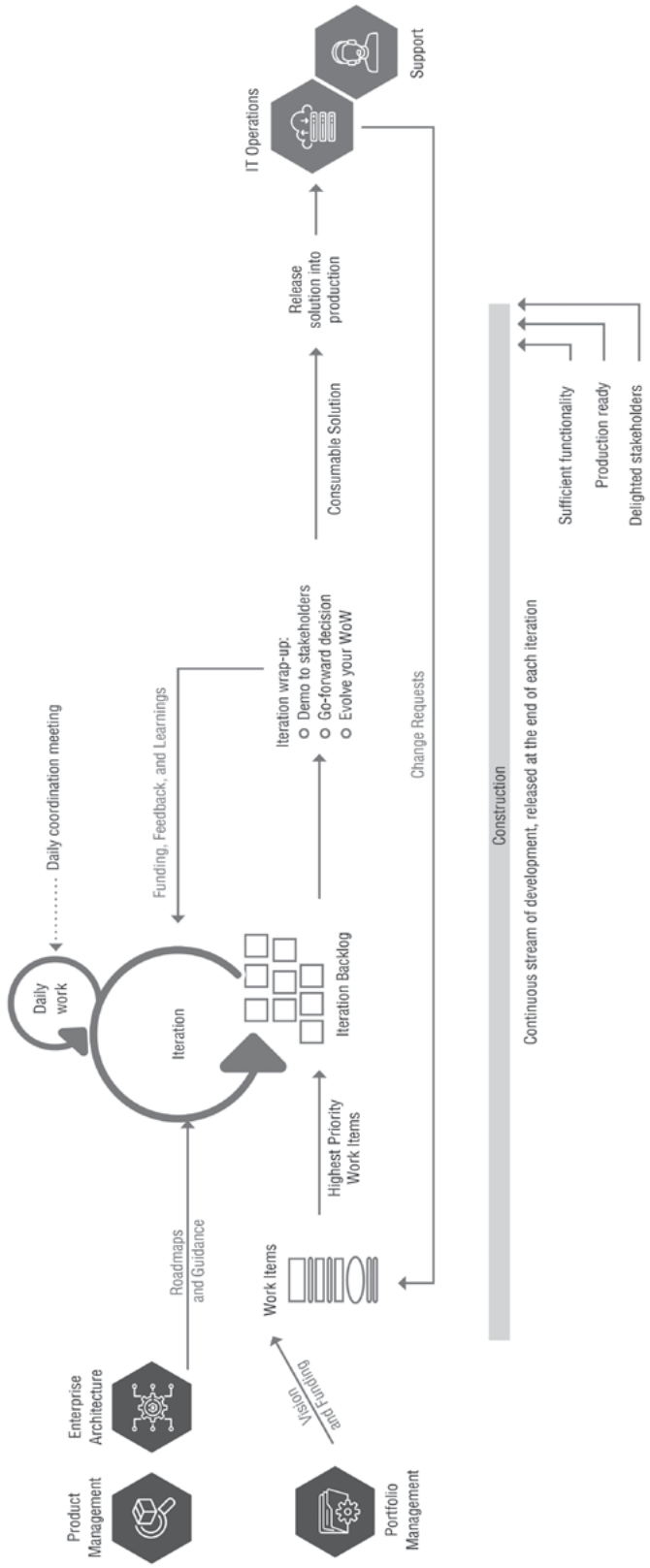


Figure 3.5 DAD's Continuous Delivery: Agile life cycle.

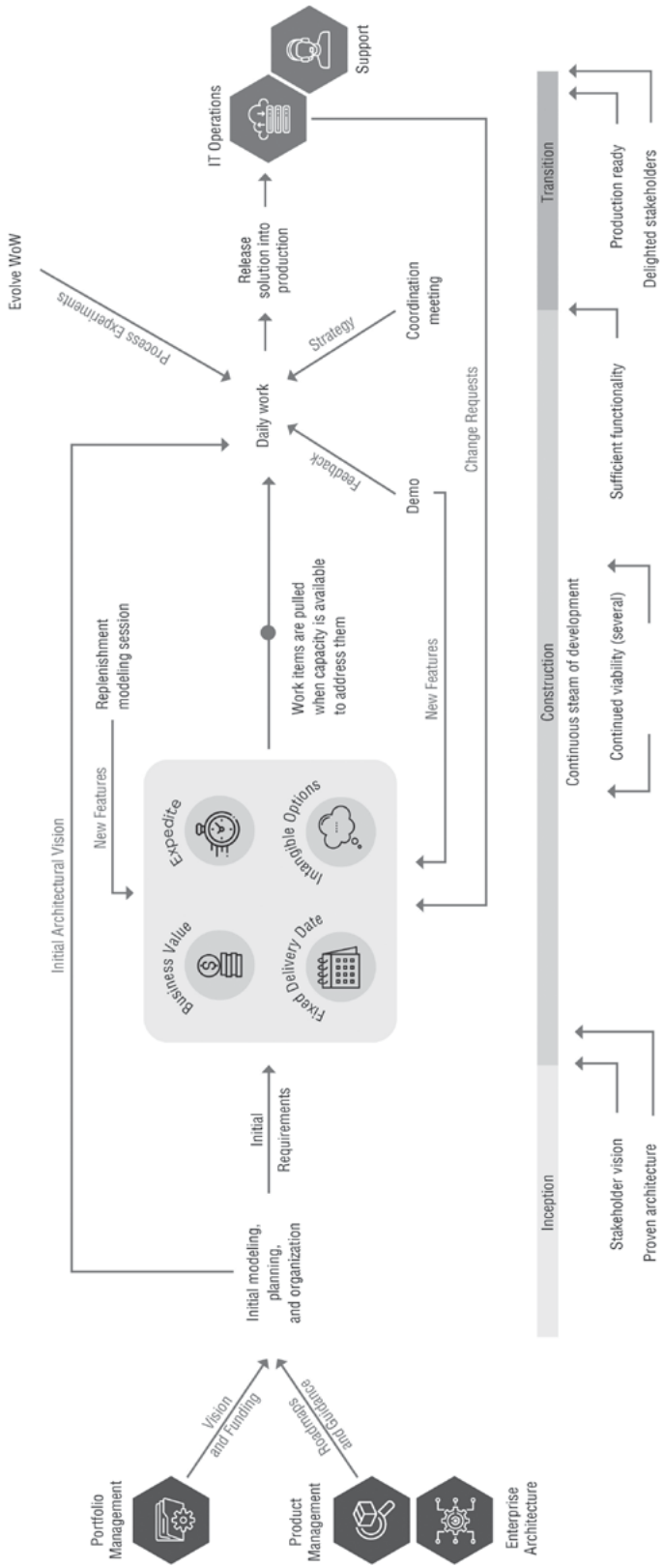


Figure 3.6 DAD's Lean life cycle.

Note that Scrum prescribes the use of a set of "ceremonies," such as the daily coordination meeting (Scrum), iteration (sprint) planning sessions, and retrospectives to be done on certain cadences within the iterations (sprints). Lean does not prescribe this overhead. In fact, it considers these practices to be a source of waste and instead suggests that they be done only when necessary. This requires a degree of discipline and self-awareness not usually found on teams new to agile. While the concepts of lean and the Kanban system are very easy to learn, it can be difficult to master the principles of lean flow and maximizing system throughput. It is suitable in these situations:

- Work can be broken down into very small work items of roughly the same size.
- Work is difficult to predict in advance. For example, teams that are focused on fixing defects or handling support issues are good candidates for this life cycle.
- The team favors the lean approach of minimizing batch size (which helps to reduce work in process) and reducing or eliminating any planning in advance of doing the work.
- The team is typically working on a series of small changes and then releasing them periodically in a manner such as a monthly release cycle.

Continuous Delivery: Lean life cycle. This life cycle, shown in Figure 3.7, is a natural progression from the Lean life cycle. It supports the goal of delivering increments of the solution in a more frequent manner than the other life cycles. Teams typically evolve into this life cycle from either the Lean life cycle or the Continuous Delivery: Agile life cycle. It requires a mature set of practices around continuous integration and deployment in order to be practical. It also requires the technical infrastructure and advanced DevOps practices that support this approach. It is best suited in these types of situations:

- Solutions can be delivered to stakeholders frequently and incrementally.
- New work, including both new requirements and defect reports, arrives often.
- Organizations have streamlined deployment practices and procedures.
- It is critical that solutions get value into the hands of stakeholders rapidly, even before the entire solution is complete.
- Teams have mature DevOps practices in place, including continuous integration, continuous deployment, and automated regression testing.
- The team is long-lived (stable), and is working on a series of releases over time.

Exploratory life cycle. This life cycle, shown in Figure 3.8, is based on the Lean Startup principles advocated by Eric Ries, and is extended with ideas from complexity theory to increase its effectiveness. The philosophy is to minimize up-front investments in solutions in favor of small experiments (called minimal viable products, or MVPs) that are market tested and measured early and often. As the solution is being developed, the delivery team has the opportunity to deliver what is truly required based on feedback from actual usage. It is useful in these types of situations:

- The solution addresses high-incertitude cases such as a new, unexplored market or a new product.
- The stakeholders and delivery team are very flexible in adapting the solution as it is being developed.
- You have a valid hypothesis/strategy to test with clear go/no-go criteria for when the test is over.
- You are willing to experiment and evolve your idea based on your learnings.

Program life cycle. Figure 3.9 depicts the Program life cycle, which organizes the workflow of a team of teams. Figure 3.10 shows the organizational structure to support coordination within

Figure 3.7 DAD's Continuous Delivery: Lean life cycle.

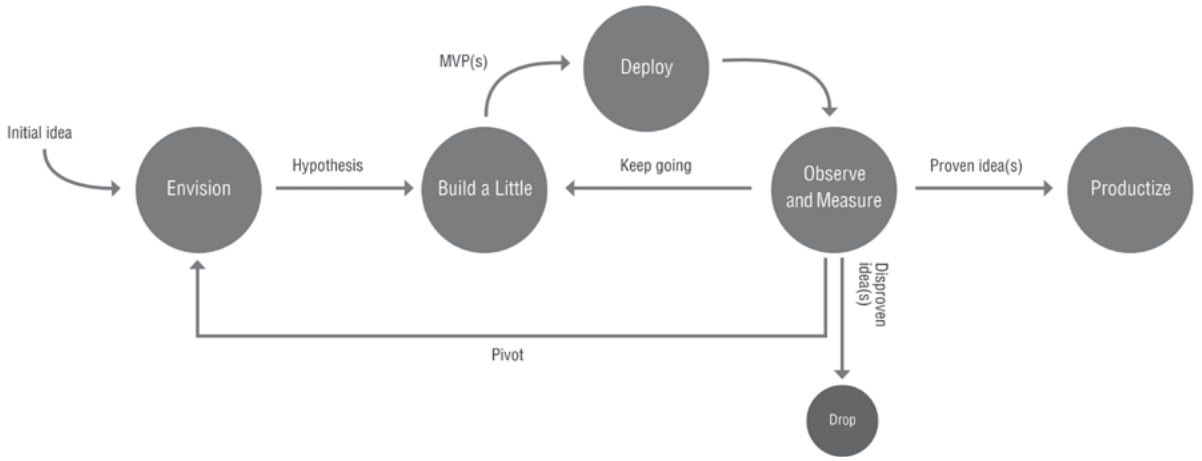


Figure 3.8 DAD's Exploratory life cycle.

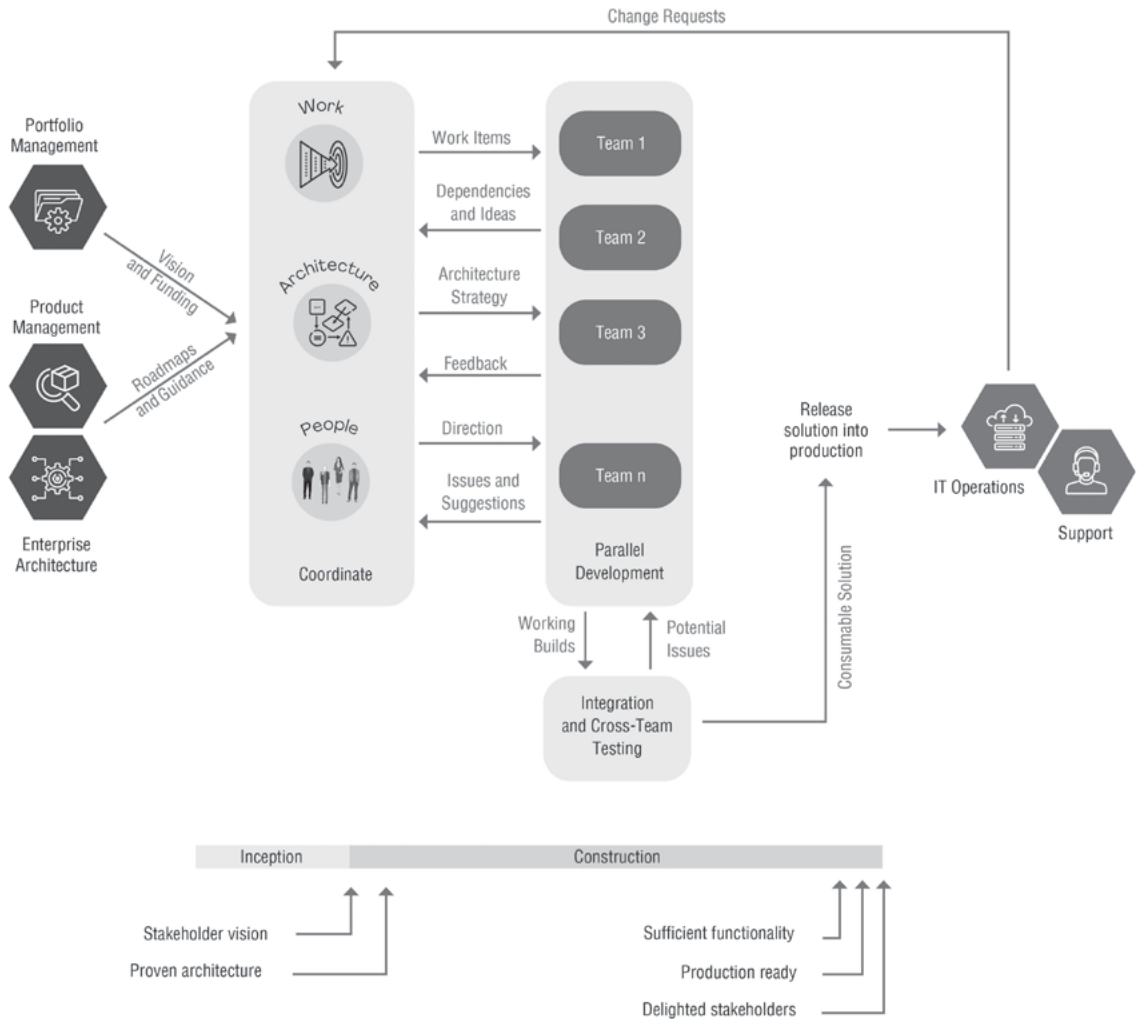


Figure 3.9 DAD's Program life cycle for a team of teams.

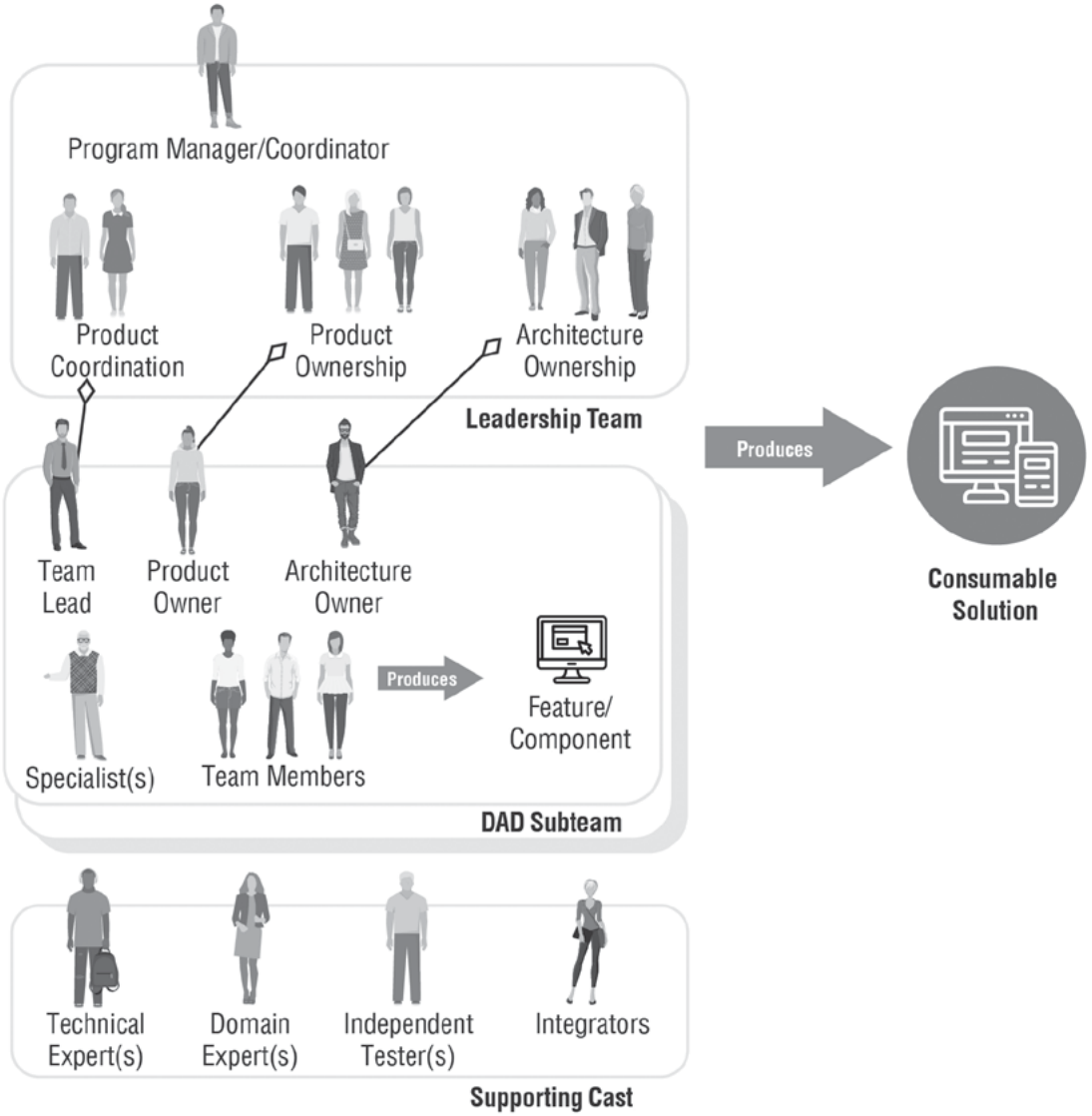


Figure 3.10 Organizing a large agile team (program).

the program. The subteams, sometimes called squads, are free to choose their own way of working (WoW), including the life cycle. Therefore, there may be some subteams following the Agile life cycle, some following Continuous Delivery: Lean, and so on. There is very often a subteam focused on system integration and system integration testing (SIT) running in parallel to the development subteams. Because DAD supports a selection of deployment strategies, captured by the Deploy the Solution process goal, your program may choose to follow an "agile release train" (ART) strategy or a more advanced, continuous release stream strategy (among several options). Large agile teams are rare in practice, but they do happen. This is exactly the situation that scaling frameworks such as SAFe, LeSS, and Nexus address. We've added an explicit Program life cycle

to make it clear how DAD scales to support large teams. The Program life cycle is useful in these types of situations:

- You have a very complex problem to address, requiring a large team to accomplish the work.
- Your solution is a platform or collection of related solutions (often called a suite).
- You are not able to reorganize the complex problem or platform/suite into smaller, decoupled solutions.

In the discussion above, we made it clear that teams often evolve from one life cycle to another. This is because Disciplined Agile teams are always striving to optimize flow, to improve their process and work environment as they learn through their experiences and purposeful experimentation. Figure 3.11 shows common evolution paths that we've seen teams go through. Along the x-axis, you see that over time teams will progress toward a continuous release strategy (along the lines of Disciplined DevOps), toward being long-lived (stable) teams as opposed to short-term project teams, toward shorter feedback cycles (due to improved collaboration), and toward automated regression testing.

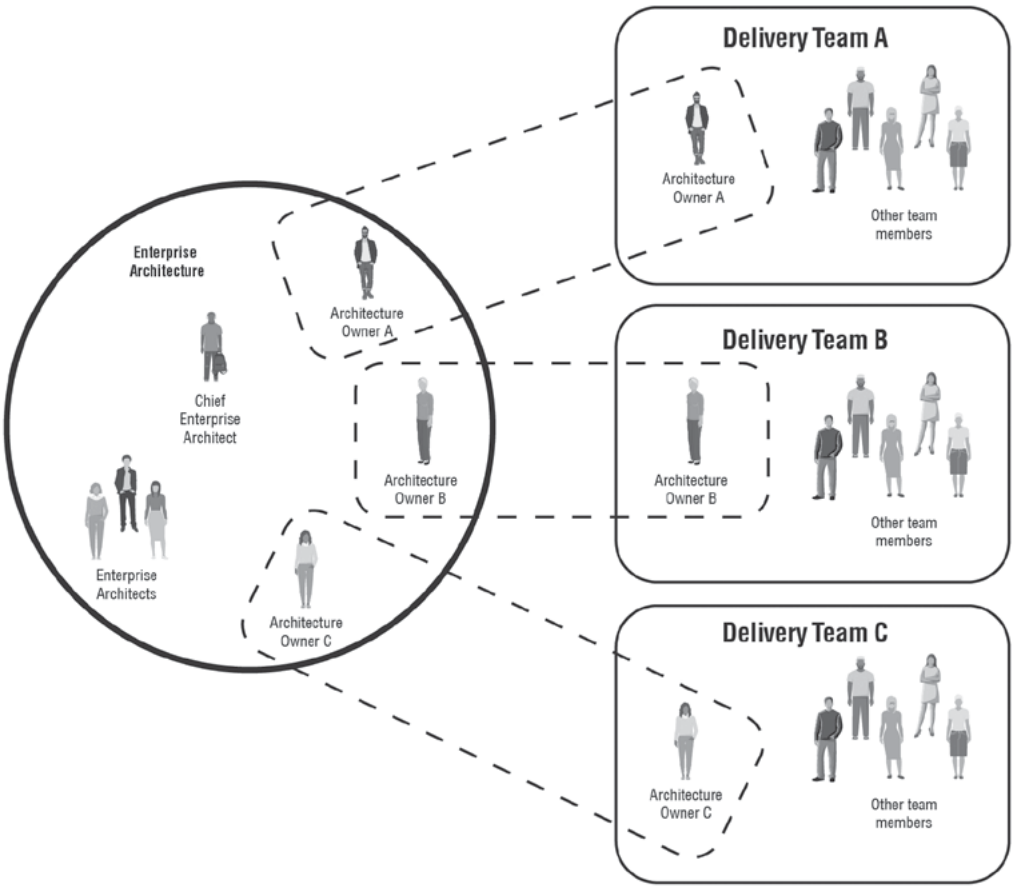


Figure 3.11 Evolution paths for adopting DAD life cycles.

What Figure 3.11 doesn't show is how the Exploratory life cycle fits in. This is because the Exploratory life cycle isn't a full-solution development life cycle in its own right. It is typically used to test out a hypothesis regarding a potential marketplace offering, and when the idea has been sufficiently fleshed out and it appears the product will succeed, then the team shifts into one of the delivery life cycles of Figure 3.11. In that way, it replaces a good portion of the Inception phase efforts for the team. Another common scenario is that a team is in the middle of development and realizes that they have a new idea for a major feature that needs to be better explored before investing serious development effort into it. In that case, the team will shift into the Exploratory life cycle for as long as it takes to either flesh out the feature idea or to disprove its market viability.

Context Counts: Disciplined Agile Teams Are Goal Driven

DAD's goal-driven approach enables DAD teams to avoid being prescriptive, thereby allowing them to be more flexible with the ability to scale more easily than with other agile methods. For example, where Scrum prescribes a value-driven product backlog approach to managing requirements, DAD instead says that during construction you have the goal of addressing changing stakeholder needs. DAD then indicates that there are several issues surrounding that goal that you need to consider, and there are several techniques/practices that you should consider adopting to do so. DAD goes further and describes the advantages and disadvantages of each technique and in what situations they are best suited. Yes, Scrum's product backlog approach is one way to address changing stakeholder needs, but it isn't the only option, nor is it the best option in many situations.

In the first DAD book, we described goals in a nonvisual manner using tables that explore the advantages and disadvantages of the techniques associated with an issue. After that book was published, we expanded on this approach and developed a way to represent goals in a visual manner using what we call a process goal diagram.

Let's work through an example. Figure 3.12 depicts the goal diagram for Explore Scope, a goal that you should address at the beginning of the life cycle during the Inception phase (remember, DAD promotes a full-delivery life cycle, not just a construction life cycle). Where some agile methods will simply advise you to populate your product backlog with some initial user stories, the goal diagram makes it clear that you might want to be a bit more sophisticated in your approach. What level of detail should you capture, if any? How are you going to explore potential usage of the system? Or the UI requirements? Or the business process(es) supported by the solution? Default techniques, or perhaps more accurately suggested starting points, are shown in bold italics. Notice how we suggest that you likely want to default to capturing usage in some way, basic domain concepts (for example, via a high-level conceptual diagram) in some way, and

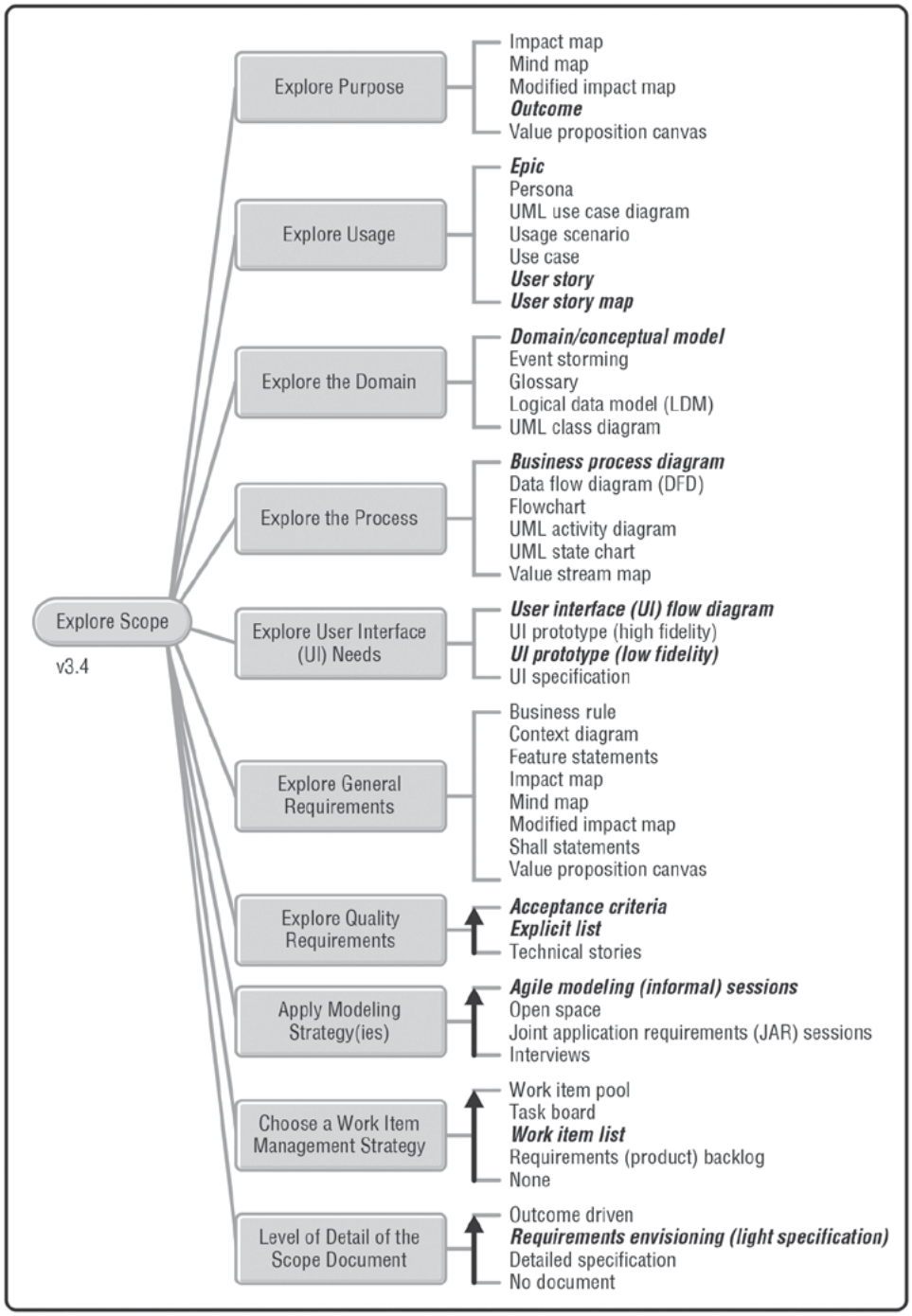


Figure 3.12 The goal diagram for Explore Scope.

nonfunctional requirements in some way. There are different strategies you may want to consider for modeling. You should also start thinking about your approach to managing your work—a light specification approach of writing up some index cards and a few whiteboard sketches is just one option you should consider. In DAD, we make it clear that agile teams do more than just implement new requirements, hence our recommendation to default to a work item list over Scrum's simplistic requirements (product) backlog strategy. Work items may include new requirements to be implemented, defects to be fixed, training workshops, reviews of other teams' work, and so on. These are all things that need to be sized, prioritized, and planned for. Finally, the goal diagram makes it clear that when you're exploring the initial scope of your effort, you should capture nonfunctional requirements—such as reliability, privacy, availability, performance, and security requirements (among many)—in some manner.

There are several fundamental advantages to taking a goal-driven approach to agile solution delivery:

- It provides a concise, shared pathway to leaner, less wasteful process decisions.
- It supports process tailoring by making process decisions explicit.
- It enables effective scaling by guiding you through tailoring your strategy to reflect the realities of your scaling factors.
- It makes your process options very clear and thereby makes it easier to identify the appropriate strategy for the situation you find yourself in.
- It takes the guesswork out of extending agile methods and thereby enables you to focus on your actual job, which is to provide value to your stakeholders.
- It makes it clear what risks you're taking on and thus enables you to increase the likelihood of success.
- It hints at an agile maturity model (this may not be a benefit).

But This Is so Complicated!

Our strategy with the DA tool kit is to explicitly recognize that software development (and IT and organizations, in general) is inherently complex. DA doesn't try to dumb things down into a handful of "best practices." Instead, DA explicitly communicates the complexity that you face, the options that you have, and the trade-offs that you're making, and simplifies the process of choosing the right strategies that meet your needs.

So yes, there are a lot of process goals (21, in fact) depicted in Figure 3.13. Which would you take out? We've seen teams not address risk mitigation in any way, thereby ignoring the Address Risk goal, but that invariably went poorly for them. We've also seen teams choose not to address the goal Improve Quality, only to watch their technical debt increase over time. The point is that, in practice, you can't safely choose to ignore any of these goals. Yes, it's daunting that there is so much to take into account to be successful at solution delivery long term, but what we've captured appears to be a minimal set.

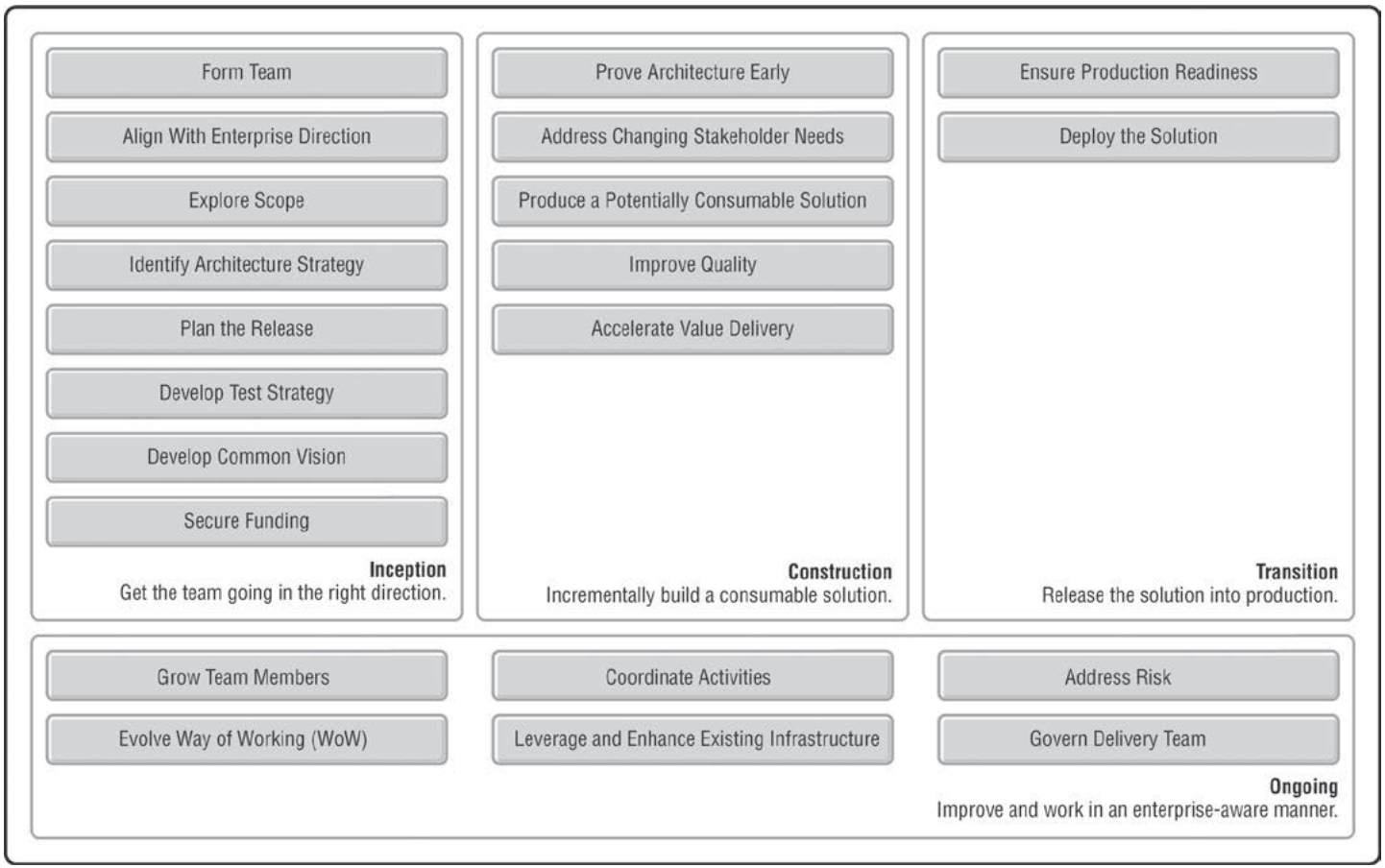


Figure 3.13 The DAD process goals.

Figure 3.13 summarizes the goals for a DAD team grouped by the three phases of Inception, Construction, and Transition, as well as the goals that are ongoing throughout the life cycle. All of the DAD process goals are described in detail in the book *Choose Your WoW!*[1]

DAD Provides the Foundation for Scaling Agile Tactically

The Disciplined Agile (DA) tool kit distinguishes between two types of "agility at scale":

1. **Tactical agility at scale.** This is the application of agile and lean strategies on individual Disciplined Agile Delivery (DAD) teams. The aim is to apply agile deeply to address all of the complexities appropriately, what we call scaling factors.
2. **Strategic agility at scale.** This is the application of agile and lean strategies broadly across your entire organization. From an IT point of view, this includes Disciplined DevOps and the overall value stream in general. From an enterprise point of view, this includes all divisions and teams within your organization, not just your IT department.

Let's examine what it means to tactically scale agile. When many people hear "scaling," they often think about large teams that may be geographically distributed in some way. This clearly happens, and people are clearly succeeding at applying agile in these sorts of situations, but there's often more to scaling than this. Organizations are also applying agile in compliance situations, either regulatory compliance that is imposed upon them or self-selected compliance (such as CMMI, ISO, and ITIL). They are also applying agile to a range of problem and solution complexities, even when multiple organizations are involved (as in outsourcing). Figure 3.14 summarizes the potential tactical scaling factors that you need to consider when tailoring your agile strategy.

DAD provides a better foundation for tactically scaling agile in several ways:

- DAD promotes a risk value life cycle—attacking the riskier work early in an endeavor in order to help eliminate some or all of the risk, thereby increasing the chance of success. Some people like to refer to this as an aspect of "failing fast," although we like to put it in terms of learning fast or, better yet, succeeding early.
- DAD promotes self-organization enhanced with effective governance, based on the observation that agile teams work within the scope and constraints of a larger, organizational ecosystem. As a result, DAD recommends that you adopt an effective governance strategy that guides and enables agile teams.
- DAD promotes the delivery of consumable solutions over just the construction of working software. In addition to producing software, DAD teams also create supporting documentation, upgrade and/or redeploy the hardware the software runs on, potentially change the business process around the usage of the system, and may even motivate changes to the organizational structure of the people using the system.

---

[1] Ambler, S., & Lines, M. (2020). *Choose your WoW! A Disciplined Agile Delivery handbook for optimizing your way of working (WoW).* Project Management Institute.

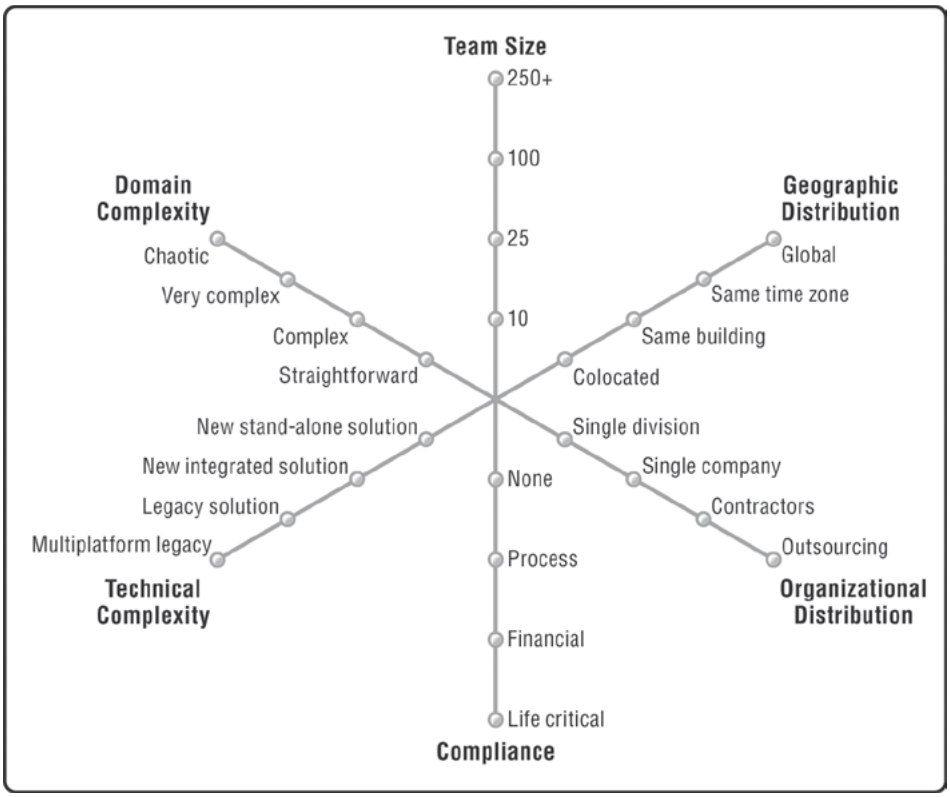


Figure 3.14 Tactical scaling factors.

- DAD promotes enterprise awareness over team awareness (this is a fundamental principle of the DA tool kit). What we mean by this is that the team should do what's right for your organization—work to a common vision, leverage existing legacy systems and data sources, and follow common guidelines—and not just do what's convenient or fun for them.
- DAD is context sensitive and goal driven, not prescriptive (another DA principle is that choice is good). One process size does not fit all, and effective teams tailor their strategy to reflect the situation in which they find themselves.

In Summary

Disciplined Agile Delivery (DAD) provides a pragmatic approach for addressing the unique situations that teams find themselves in. DAD explicitly addresses the issues faced by enterprise agile teams that many agile methodologies prefer to gloss over. This includes how to successfully initiate agile teams in a streamlined manner, how architecture fits into the agile life cycle, how to address documentation effectively, how to address quality issues in an enterprise environment, how agile analysis techniques are applied to address the myriad of stakeholder concerns, and many more.

Chapter 4

Introduction to the Case Study

We are now going to describe what you might expect on a typical DAD team using a fictitious case study based on a retail bank called "BigBank." The company would like to develop a solution to enable prospective customers to apply for a mortgage online. The solution should be easy to use on both a desktop browser and mobile responsive devices.

After trying Scrum on several teams, BigBank realized that they needed to adopt an agile method that reflects the enterprise challenges they face. In the words of one of their senior developers, "My team doesn't have the time to figure out all the stuff that Scrum doesn't address." After a bit of investigation, they chose to adopt DAD because it offered the flexibility they required. BigBank liked the fact that DAD includes a life cycle that extends and enhances Scrum's life cycle, allowing them to retain the investment that they've made in Scrum. Given DAD's lack of prescription and focus on pragmatism, BigBank recognized that DAD would scale to meet its diverse needs.

As described previously, DAD provides guidance for the hundreds of decisions made on a typical team. For the purposes of this case study, we will only discuss a small number of decisions with the intent of illustrating how DAD provides lightweight guidance for the entire delivery life cycle. Admittedly, in the interest of brevity, this is an oversimplification of a typical delivery team.

The Team

Let's introduce the team.

- **Patricia, the product owner:** Patricia works for Berhard, the primary business sponsor. She has no technical background but has been asked to be the product owner for this team. She is concerned about having to work full time in the same work area as the rest of the development team.
- **Barbara, the business analyst:** Barbara is a consultant who brings many years of experience with modeling and elicitation of requirements and business process design. She reports to Patricia.
- **Terry, the team lead:** Terry was previously a Certified ScrumMaster® (CSM®), but recently took a DAD workshop that opened his eyes to responsibilities in an enterprise situation that extend beyond just those of scrum master. As such, he took it upon himself to lead the rest of the team.
- **Carlos, the coach:** Carlos is a Green Belt-certified DAD coach and draws upon his years of experience seeing patterns of both success and failure on agile teams. He is coaching several DAD teams besides this one.

- **Ashok, the architecture owner:** Although Ashok is definitely a senior developer, he realizes that being an architecture owner includes responsibilities beyond being the smartest and most experienced developer on the team.
- **Danny, the developer:** Danny is a junior developer who has been with BigBank for two years. Danny is very excited to be on an agile team.
- **Debbie, the developer:** Debbie is an intermediate developer who has worked on web-based systems her entire career. She isn't sure about this new agile approach, but is willing to give it a try.
- **Tara, the tester:** This is Tara's first agile team. She has always applied traditional testing practices on previous teams.
- **Dick, the database administrator:** Dick has always worked on teams that do complete requirements and design at the beginning of the life cycle. He expects that the database logical and physical designs will be completed and reviewed by his group before any coding begins. Terry has explained to him that he will need to take an evolutionary approach, which he is not too happy about. Although Dick is only available to the team 25% of the time, it is his highest priority.

Now, let's introduce the primary stakeholders:

- **Victoria, the vice president of IT:** Victoria has asked this team to pilot the DAD approach and is interested to know how it can help deliver the promise of agile that they have not been seeing from their adoption of Scrum.
- **Berhard, the business stakeholder and sponsor:** Although Berhard is sponsoring this team, he has many other responsibilities.
- **Chris, the Certified ScrumMaster® (CSM®):** Chris is *not part of the team,* but is an interested observer. He takes a "no compromise" approach to Scrum and is not sure why we would need DAD in addition to Scrum.
- **Enrico, the enterprise architect:** Enrico is responsible for ensuring that all IT solutions are well designed and can support nonfunctional requirements such as security, scalability, and fault tolerance. He is worried about agile teams ignoring standards and existing assets, such as proven services that are available for reuse.
- **Oliver, the operations manager:** Oliver and his team will have to deploy and support this application when it goes into production.
- **Samira, the support supervisor:** Samira manages the support center and is concerned about how issues will be resolved by the team after it is deployed into production.
- **Mindy, the vice president of marketing:** Mindy's team is responsible for all public-facing messaging for BigBank. Her team will need to put the marketing program together for this new customer offering.
- **Padma, the PMO lead:** Padma is responsible for reporting the health of all teams and assessing delivery risk across the IT portfolio.

The Approach

One of the very first decisions that any team needs to make is which of DAD's six life cycles to use for this team. Based on a suggestion from Terry, the team lead, the team elected to use the Scrum-based agile/basic DAD life cycle for the following reasons:

- The team is new to agile, and the basic life cycle provides sufficient structure for new teams.
- The work can be identified, prioritized, and estimated (at a high level) in advance, although requirements are still expected to evolve over time.

We pick up this case study at the beginning of the Inception phase.

Chapter 5

Inception

The Inception phase in DAD is an explicit team-initiation effort. It should be as lightweight and short as possible. You want to form your team and perform sufficient modeling and planning so that you can get going in the right direction. You do this at a high level to think through the major issues that you will face, trusting that your team will be able to handle the details as they evolve.

Team kickoff meeting: Terry schedules a meeting with the team for a team kickoff. They gather on Monday morning. Terry asks Berhard, who is the sponsor, to kick off the meeting by explaining the need for the new mortgage application portal (MAP), and how it is a critical business initiative to attract new mortgage customers to the bank. After Berhard answers several questions from the team, Terry describes the strategy for developing the MAP solution using DAD. He explains to the team that he has timeboxed three weeks for the Inception phase, which in Scrum is often referred to as "Sprint 0." While he expects that future releases of the solution will require shorter Inception phases, he feels that it is prudent to start with a longer Inception because the team is new to DAD and training will be part of this phase. The purpose/milestone for this phase is to obtain agreement that the vision makes sense from the perspective of all stakeholders. He explains that this means that they need to do "just enough" work to define things that you might find in a typical project charter, albeit in a lightweight fashion. Using a projector, Terry reviews the goals for DAD's Inception phase.

Terry: Terry says that it looks like the team is well on the way to achieving these goals. They already have formed the initial team. As part of developing a common vision, they will work on the other goals shown in parallel. The team begins discussing their strategy for the next three weeks to gather the information necessary to achieve these Inception goals:

- **Align With Enterprise Direction:** Terry asks Ashok to schedule a meeting with Enrico to obtain information on any existing architecture assets, patterns, standards, and guidelines that the team should be aware of. He suggests that Enrico share information on the enterprise architectural vision so that the new solution is aligned properly.
- **Explore Scope:** Patricia plans to schedule meetings with business stakeholders (Berhard and several others) to build a prioritized backlog of agile requirements (stories). Barbara, the business analyst, will assist Patricia with facilitation of these meetings.
- **Identify Technical Strategy:** Ashok will spend some time with Terry and the other developers to model an initial architectural strategy using some lightweight modeling techniques.
- **Develop Release Plan:** Once the scope has been determined, Terry will ask that Patricia schedule a meeting to do estimation with the team so that a high-level release plan can be determined.
- **Secure Funding:** Berhard has already decided to fund the three-week Inception phase and has tentatively set aside funding for the next four months for the Construction and Transition phases.

He understands that the funding level may need to be negotiated based on the findings during Inception.
- **Develop Test Strategy:** Once the nonfunctional requirements and a reasonable technical architecture are initially identified, the team works through their test strategy for MAP. They identify how they will work together to test their system, what tools they hope to use, and what their testing environments will be.

The team realizes that they will need to execute on two important ongoing goals during Inception:

- **Evolve Way of Working (WoW):** While the team is currently scattered across three floors of the building, Terry wants to move the team to a common work area. He plans to work with the facilities team to establish a new workspace. The team also begins to set up their tools (their virtual work environment).
- **Address Risk:** Terry asks Ashok to work with him on creating a list of risks with mitigation strategies. Many of these risks are expected to be technical in nature, depending on the technical strategy selected, as well as the nonfunctional requirements that will be fleshed out as part of exploring the initial scope.

Week One of Inception

To ensure that everyone is on the same page and is familiar with both agile fundamentals and DAD, the team attends a two-day Disciplined Agile Lean Scrum Master (DALSM) workshop.[1] On Thursday, they jump in and start a series of workshops to do the work required to meet the Inception milestone of attaining stakeholder agreement on the solution vision. For most of these meetings, the core team will participate, along with other interested stakeholders, depending on the type of workshop. Carlos will attend the meetings to help facilitate the workshops. Coaching a team is particularly useful following a workshop to engrain the concepts learned.

Patricia works on identifying the initial scope. Patricia meets with the business stakeholders to start work on identifying the initial scope. Referring to the Explore Scope goal diagram from DAD, the team decides to do some usage modeling to flesh out the high-level requirements. They choose to use a story map,[2] see Figure 5.1, to organize the system flows into high-level epics and stories, each of which is written on a separate index card. The epics are written on orange cards, and the stories on yellow. On the backs of the cards is a set of high-level acceptance criteria that describe what conditions need to be met for the story on each card to be considered complete by Patricia (the product owner). Many of the stories will be too big to implement in a single iteration and will need to be broken down into smaller stories, but at this point they are sufficient for conveying the high-level initial scope.

Ashok leads the team in identifying the initial technical strategy. Terry and the developers meet among themselves to evaluate potential architectural strategies. They refer to BigBank's technical roadmap, enterprise standards, and reference architecture documentation to see if there are existing assets such as templates, services/APIs, and data sources that they can reuse. As suggested by DAD's Identify Architecture Strategy goal, they choose to keep the level of detail at a high level,

[1] DA course offerings can be found at PMI.org/what-is-disciplined-agile

[2] See Patton, J. (2014). *User story mapping*. O'Reilly Media.

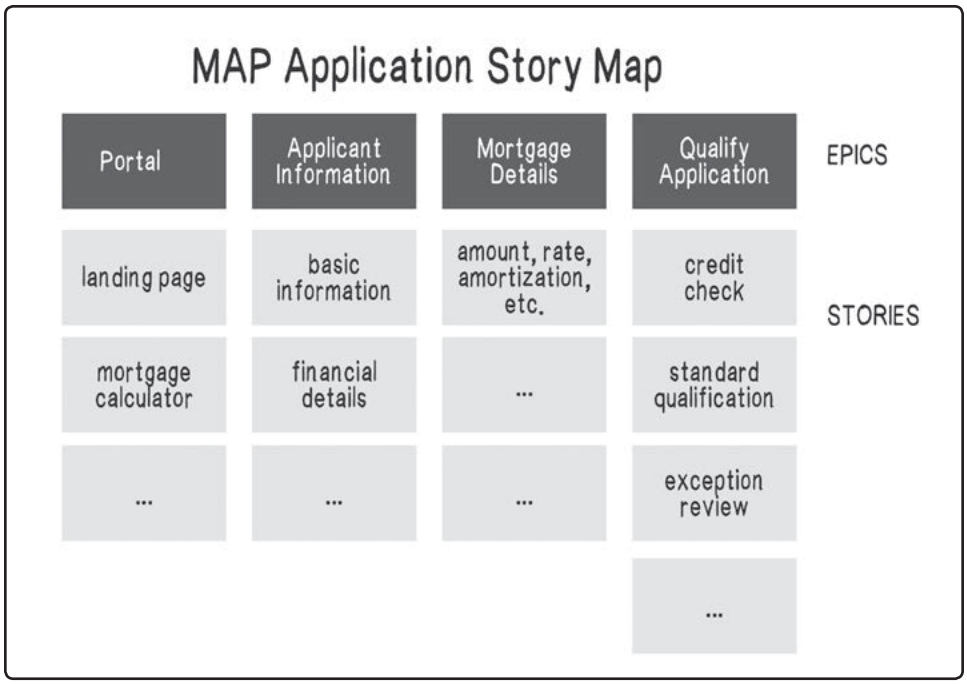


Figure 5.1 Story map for the application.

gather their ideas via informal modeling sessions, and focus on a single candidate architecture. The architecture envisioning workshop takes a full day. The team is able to find a large meeting room to do this work that has a large table and a lot of wall space. Unfortunately, there was only one small whiteboard at the front of the room. Terry orders several rolls of "whiteboard wallpaper" that can be attached to the wall via static cling. After taking down several framed motivational pictures, they use the wallpaper to turn the side walls into workable whiteboards.

Terry conveys to the team that Enrico mentioned that the enterprise architecture (EA) group wants to move away from the current monolithic architecture to a microservices model over time. They would also like to use "feature toggles" to be able to share all code on one mainline branch rather than working in separate feature branches, controlling what is actually deployed using these toggles. This will provide isolation of deployed functionality so that there is no single source of failure that could bring the whole system down. This will reduce deployment risk in the future and facilitate smaller deployments on a much more frequent basis. Feature toggles are a common DevOps strategy. BigBank's technology roadmap, created by the enterprise architecture team, defines this strategy. New functionality will be built along the microservices/feature toggle strategy. Existing legacy code and data sources will be refactored over time on an as-needed basis.[3]

During the architecture envisioning workshop, Dick brings up the need to create a comprehensive data model before Construction can begin. Carlos points out that with an agile approach, the detailed design emerges over the course of development. At this point in time, the team only needs to invest

[3] For practical strategies, see Feathers, M. (2004). *Working effectively with legacy code.* Prentice Hall; and Ambler, S., & Sadalage, P. (2006). *Refactoring databases.* Addison Wesley.

enough modeling effort to get going in the right direction. The team has done enough requirements and architecture modeling to drive the planning efforts that will occur next week. Dick is clearly not comfortable with this strategy, but agrees to wait and see how well it works in practice.

One of the things that the mortgage application portal (MAP) must be able to do is determine whether someone is an existing BigBank customer. Ashok points out that BigBank's technology roadmap calls out a collection of reusable web services that will do that and more. These web services are currently under development, and the basic customer information access services should be available two months from now. These services are being developed by "Team X-Ray" following a traditional, waterfall-style approach. Dick says that it would be easier to just write simple Structured Query Language (SQL) statements in their code to access the required legacy data. Ashok agrees that would be an easier approach, but that it would only be a short-term solution that would increase the overall technical debt within BigBank, because yet another application would be directly coupled to the legacy data sources. Carlos suggests that the team should work in a more disciplined, enterprise-aware manner that reflects the overall strategy of BigBank. A bit of extra work now will result in a more robust solution, although it increases the risk that the team faces as they now depend on Team X-Ray to deliver the required web services.

At the end of the day, the team has a set of technology and user interface free-form sketches that overview an architectural strategy that they believe will work. The diagrams are captured with a smartphone and stored in the team wiki in the architecture handbook section.

Week Two of Inception

Patricia meets with the business stakeholders to prioritize the work. Patricia reviews the story map with the other business stakeholders. They have decided to use DAD's work item list strategy as a backlog of work that will include the stories related to functionality and other work required to produce the solution. They prioritize the cards into a stack representing the work according to the highest business value. Patricia explains to the team that these priorities may need to change somewhat based on the advice of Ashok, the architecture owner.

Terry, Patricia, and Ashok meet to identify an initial list of risks. Terry and Ashok question Patricia about the nonfunctional requirements for this solution. How many people will be using this application, during what hours, and what are the peak activity periods? How many transactions per hour does she expect? She continues and asks many other questions about reliability, security, performance, and scalability. These nonfunctional requirements are captured in a list on the whiteboard, but eventually transcribed onto a wiki page that is available to the entire team to reference and update as needed.

Based on the newly identified nonfunctional requirements, the team evolves their architectural vision and discusses what aspects of the architecture could be the "trickiest" to implement. They capture the key technical risks faced by the team and brainstorm ways in which they can mitigate these risks. Ideally, mitigating the risk by implementing a piece of functionality related to the risk is preferred, but sometimes a technical proof of concept (PoC) known as a "spike" might be used.

Ashok and Patricia meet to review the prioritized work item list. With his risk list in hand, Ashok meets with Patricia to review the work priorities. He identifies some parts of the solution that he suggests be moved up to a higher priority in order to mitigate the key technical risks. The strategy is to build an end-to-end working skeleton of the solution that implements these risky, yet still important, requirements early so as to prove the architecture with working code. This will help to remove a lot of the technical risk from the effort early on during Construction, thereby increasing the team's chance of success. Surprisingly, DAD's common-sense approach to mitigation of risk early is not an explicit part of Scrum, often resulting in unpleasant surprises late in the life cycle.

Patricia and the development team conduct sizing of the work item list. After considering the strategies in the Plan the Release goal diagram of Figure 5.2, the team elects to do estimating with the planning poker technique. Since they do not have a deck of planning poker estimation cards, they decide to use a planning poker app on their smartphones. The team is already familiar with how to do estimation this way from their DAD workshop last week. They methodically work their way through the stories, with the product owner describing each story and its acceptance criteria, and subsequently the development team provides an estimate for each. Since this is the first real-world application of this technique for the team, it initially takes some time.

After three hours, they are 20% through sizing the stories. Although they have sped up with experience, it's clear that planning poker will take two days in total at their current pace. Carlos suggests a technique called relative-mass sizing. Because the team has gotten to the point where they have a common understanding of how to size things, this strategy is now an option for them. They clear off their boardroom table and, using masking tape (painter's tape), they section it off into six "size sections," in this case 1, 2, 3, 5, 8, and ?. The "?" section is for stories that are either larger than eight (and will be resized later via planning poker), something that the team needs to discuss further, or something that needs to be reorganized into smaller stories. Each person grabs a subset of the story cards and begins placing them in the size section that they feel is most appropriate. As they're doing this, they are also watching how others are placing cards. When they see something that they don't agree with (e.g., someone else has indicated that a story should be a size three, but they think that it's a five), then they say something along the lines of "Who thinks this is a three?" and the two people talk it through. The stories are sized by the team in parallel, speeding things up. The team spends an hour and a half sizing the stories, ending up with five that they felt needed to be explored in greater detail. Patricia agrees to work with the stakeholders to flesh out those stories further.

The team creates a release plan. The team decides that they will go with two-week iterations for the Construction phase. The total relative points for all the work that they estimated summed to 160 points. The team is uncertain as to how many iterations they should plan for the release in order to complete this work. Carlos explains an approach that he has seen work well on other teams. On an agile team, we certainly do not want to invest weeks doing detailed estimates on all of the work, but without some idea of what can be completed in an iteration, it is difficult to plan. He suggests that the team invests a half day to plan the work for the first iteration of Construction. After decomposing a set of work items/stories that are planned for the first iteration into detailed

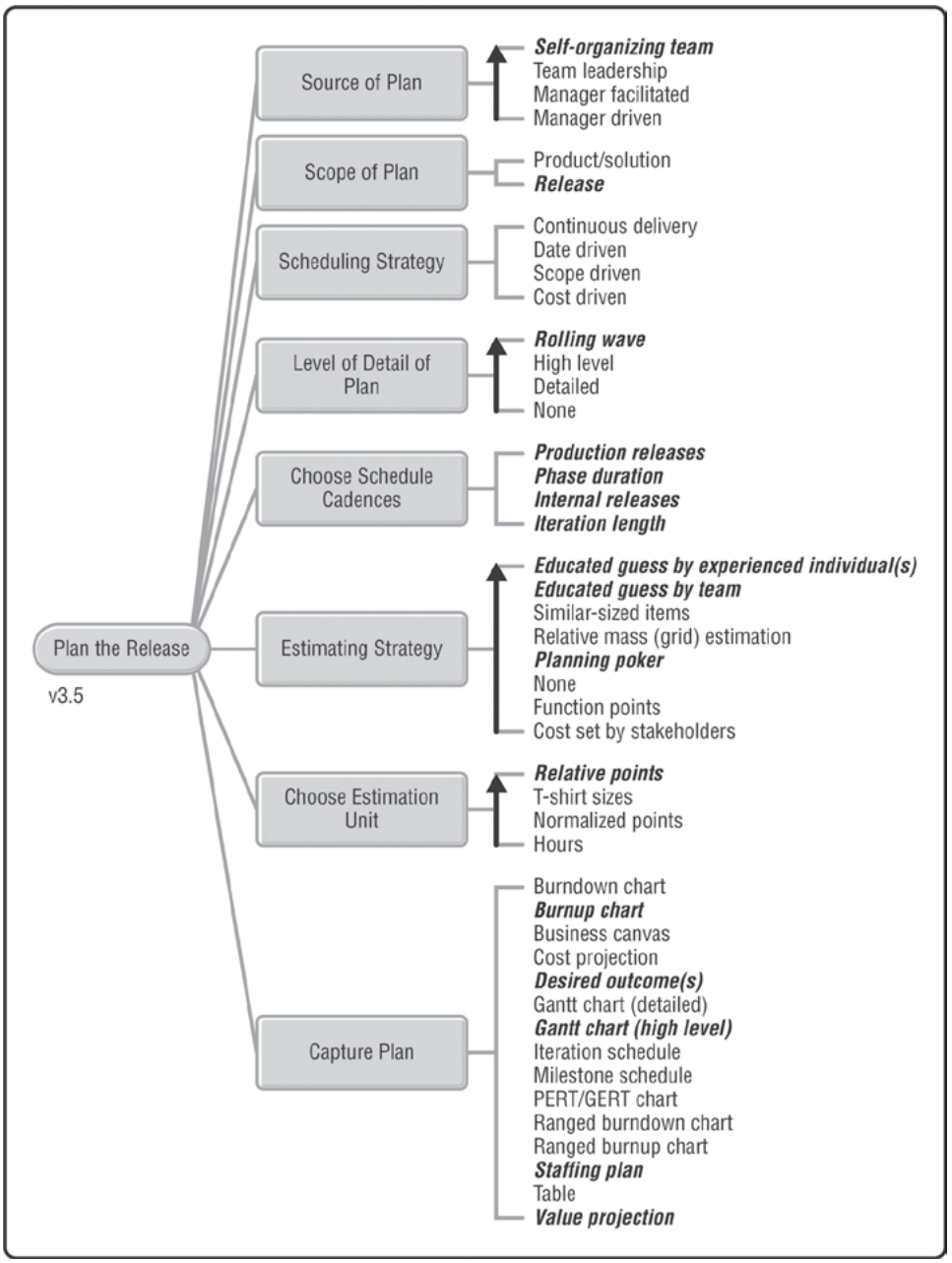


Figure 5.2 The process goal diagram for Plan the Release.

tasks and estimates, and then doing a sanity check to ensure that the team has enough hours/capacity to complete the work in two weeks, the team can gauge how many points can be delivered in an iteration. Carlos warns Patricia that it is only a guess since it is difficult to estimate without actual experience based on this team, technology, and other situation-dependent factors. The team decides to spend the next morning estimating.

Estimating the number of iterations required. Based on the detailed planning session, the next morning the team determines that they can complete the five highest priority items from the work

item list. These items represent estimates of 20 total relative points. Based on a total of 160 points for the entire backlog, and the expected team's "velocity" of 20 points per iteration, simple division would indicate that it will take eight iterations to complete the application. Carlos suggests—considering the degree of uncertainty related to a new team, technology, and process—that some contingency should be added to the release plan. He suggests that two iterations be added to bring the total to 10. This allows for unexpected "surprises," including possible scope additions that should not jeopardize the ship date of the application. They agree that the Construction phase will be 10 iterations of two weeks each. This identifies a date that the business stakeholders can depend on. The greatest frustration that Berhard and other business stakeholders have with IT is their inability to deliver on time. One of the reasons that DAD has been adopted is to address this problem.

As you can see in the goal diagram for Plan the Release, an alternative approach would be to set the expectation for the system to be released as soon as sufficient functionality is available. This is called a scope-driven approach versus the date-driven approach selected by the team.

Carlos then suggests that the team members agree on the time required to transition to the stakeholders. DAD recognizes that stakeholders include more than just end users of the application. Groups that will be responsible for supporting the solution in production are also stakeholders, auditors, finance people, and many more. Carlos advises that the Transition phase be extended beyond the go-live date to allow for a reasonable handoff period to the support group. They agree that a three-week Transition phase should be sufficient. The release plan also shows three weeks of time for "usage" of the application that BigBank typically calls a warranty period after expected completion. DAD has a milestone called Delighted Stakeholders to signify completing the release.

Terry creates a high-level Gantt chart, see Figure 5.3, to show the release schedule and associated milestones. This schedule will be updated periodically throughout Construction based on actuals, and will be put on the wall as an information radiator.

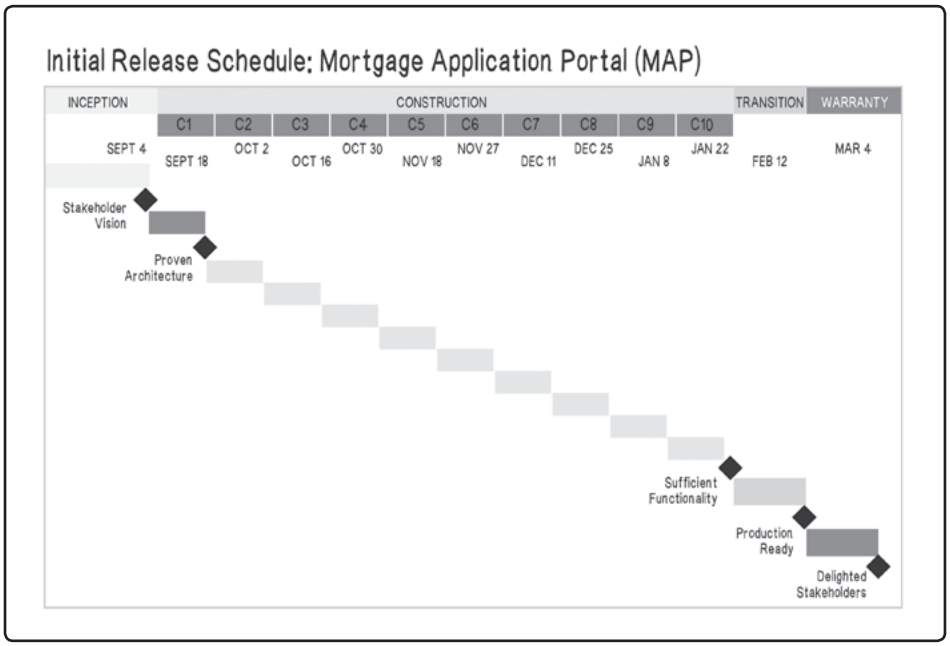


Figure 5.3 Initial release schedule for the MAP team.

Week Three of Inception

Terry works with the facilities group to set up a common work area for the team called a team room. During the last two weeks, Terry and Carlos have been scouting the building for a good work area in which to colocate the team. They secure a 24×24-foot area, which they have christened the "agile pod." During this week, they worked with the development team and Patricia to organize desks in a manner that fosters team collaboration. Carlos suggested several layouts based on what he has seen work in other organizations. The figure here shows what the new work area looks like. The outer walls are high cubicle walls, roughly 6.5 feet in height, with the exception of the walls at the top, which are about 3.3 feet in height. In the top-right corner of the pod area, there are two large whiteboards, as well as space for a manual task board. They will use colored sticky notes to show the progress of their tasks against the iteration's work items. It is in this area that the team will hold their daily standup coordination meetings. There is also a whiteboard next to it for ad hoc modeling. The two angled rectangles represent whiteboards on wheels. In the top-left area, the team has a table for team collaborations, such as iteration planning, reviews, and retrospectives. A monitor hangs on this wall so that ad hoc demonstrations to interested stakeholders can take place at any time. It is also useful for other team reviews, such as backlog reviews and estimating. The team thinks that they may wish to migrate their manual task board to a virtual board at some point, in which case the board will be displayed on this monitor. A key philosophy of creating this pod is to give the team an environment in which they will not need to leave the work area for meetings except in rare situations such as town hall meetings. This fosters team collaboration and focus, where team members are expected to be available to help each other at all times.

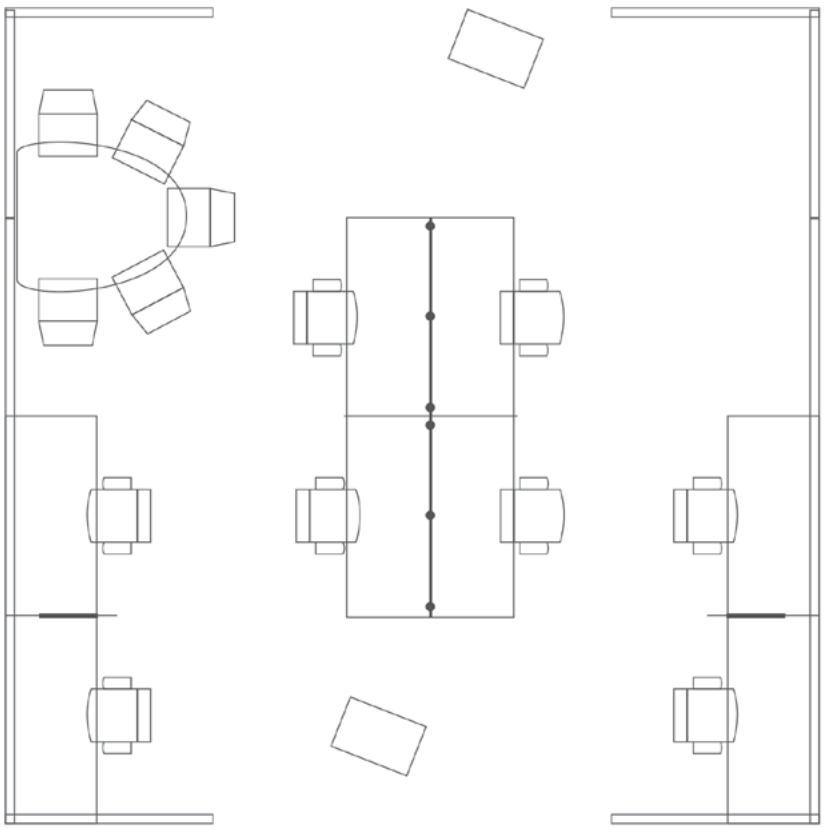


Table 5.1 Risk list for the MAP team.

Risks and Mitigation Strategies	Prob.	Impact	Sev.
Delivery date for customer web services slips • Ashok to work closely with Team X-Ray to monitor their status	60%	8	4.8
Stakeholders aren't available • Patricia to work closely with them to ensure availability • Berhard to support Patricia	30%	8	2.4
Architecture strategy fails • Will prove it in Iteration C1 with working code	10%	10	1
New versions of our supported browsers will be released • Debbie will monitor release announcements • Team will develop automated test suite	100%	1	1

Ashok works with the developers to set up the tooling environment. Beginning late in the first week, the development team starts setting up the development environment and supporting tools. For this solution, they will use a set of Java-based technologies. In the spirit of allowing the team to self-organize, Terry allows each developer to use the integrated development environment (IDE) of their choice. They install Git as their source code repository and, as a group, they develop a source code management and branching strategy. They also set up a build server to automatically create builds upon checking in code changes and to run automated developer regression tests (unit tests). While they would also like to create some scripting to allow testers to self-deploy to a separate testing environment, they run out of time in this phase. They resolve to do this in the first iteration of the Construction phase. They convince Patricia to add the requirement to create deployment scripting to the work item list as a high-order work item.

Terry leads the team to update the risk list. The team spends about 30 minutes brainstorming potential risks that they face. They capture these risks on the whiteboard initially, rating them on likeliness to occur (percentage) and the likely impact (scale of 1–10) if they do occur. Simple math (the likeliness of occurrence times the likely impact) gives a severity rating that is used to prioritize the risks. After the meeting, Terry captures the risk in a spreadsheet (see Table 5.1), which he then prints and puts on the wall beside their task board. The risk list, sorted in priority order, follows below.

Summarizing the Inception work into a vision. Patricia, Ashok, and Terry work together to consolidate their work from the last three weeks into a vision statement that they can review with stakeholders. They choose to present this work in a PowerPoint presentation. Carlos provides a template from his work on similar teams at other organizations. They create one or two slides to present each of the following topics:

- Business goals of the initiative
- Success criteria

- Key stakeholders and roles
- Vision
  - Initial scope of the effort
  - Initial technical strategy
  - Initial release plan
  - Work environment details
  - Initial risks

The Stakeholder Vision milestone review. At the end of the Inception phase, a review of the vision is conducted with key stakeholders as part of reaching the Stakeholder Vision milestone. The meeting is scheduled for one hour. The entire development team, as well as stakeholders from the business, operations, and support, attend the meeting. Additionally, other interested stakeholders attend from enterprise architecture, technical documentation, training, internal audit, and the project management office (PMO). Although such a large group might not attend each Inception milestone review, they should at least be given the opportunity to review the vision so that there are no nasty surprises later on.

Patricia, Terry, and Ashok present parts of their Inception findings. Twenty minutes are reserved for questions from the stakeholders.

Terry and Patricia conclude the meeting by asking for "general agreement," sufficient to achieve the Stakeholder Vision milestone and thus proceed into the Construction phase on Monday.

Berhard, the business sponsor, and Victoria, the vice president of IT, commend the team on their excellent work in Inception and agree that the vision, as framed, seems reasonable. Carlos reminds everyone that while the vision gets the stakeholders and the team on the same page and gets everyone going in a common direction, the vision will be revisited at the end of each iteration to ensure that things are unfolding as described in this meeting. The meeting adjourns. The team is excited about starting Construction on Monday. They agree that investing three weeks doing high-level planning and modeling was a good use of time.

The team spends the remainder of the day moving from their current desks into their new desks in the agile pod. They are genuinely excited about coming back into work on Monday morning.

Chapter 6

Construction Iteration C1

After a short, lightweight Inception phase where the team invested the time to gain agreement around the approach that they are going to take, they are ready to begin Construction. Note that it is impossible to devote as much space in this book as is needed to comprehensively cover what happened with this team. Therefore, we describe key events and issues that were encountered and the strategies that the team took to address them. The transparency built into the agile process, combined with constant self-evaluation by the team, quickly surfaces problems so that they can be fixed in a timely and open manner. Also keep in mind that this team has chosen to use DAD's Agile (Scrum-based) life cycle. For a different life cycle, the experiences would be quite different.

Day 1. Normally, the team would spend the first half of day one of the iteration conducting an iteration planning session. However, since the team invested some time in Inception to plan the first iteration to improve the quality of their estimates for release planning, the planning for this iteration is already done.

There are six stories, see Figure 6.1, which total 20 points. Three of the stories address technically significant functionality that, when implemented, will prove that the team's architectural strategy (identified during Inception) will actually work. There is a chance that the team will discover that the strategy doesn't work and, in that case, they will need to evolve their architectural approach based on their learning. This is an aspect of DAD's risk value life cycle strategy, where technical risk is considered when prioritizing the work, leading to the Proven Architecture milestone early in Construction.

They do, however, need to create their task board that you can see in Figure 6.2. On the whiteboard in the corner of their new pod, they create columns for Not Started, In Progress, Needs Validation, and Completed. Carlos points out that, in his experience, different teams customize the names and types of columns based on what they think is most effective, but these columns are good ones to start with. The team then creates sticky notes representing each work item/story and puts them in the left column going down in priority order.

They then have a coordination meeting to plan the day's work. Each team member takes one or two sticky notes for the tasks that they intend to work on today and moves them to the "In Progress" column. They ensure that these tasks have their initials on them so that it is clear who is working on each task. Adding up the total hours on all the sticky notes, the team comes up with a total of 240 hours of work to be completed over the 10-day iteration. The team understands that unexpected work will arise during the iteration, so they have ensured that the 240 hours is about 20% less than their total available hours to allow for contingency.

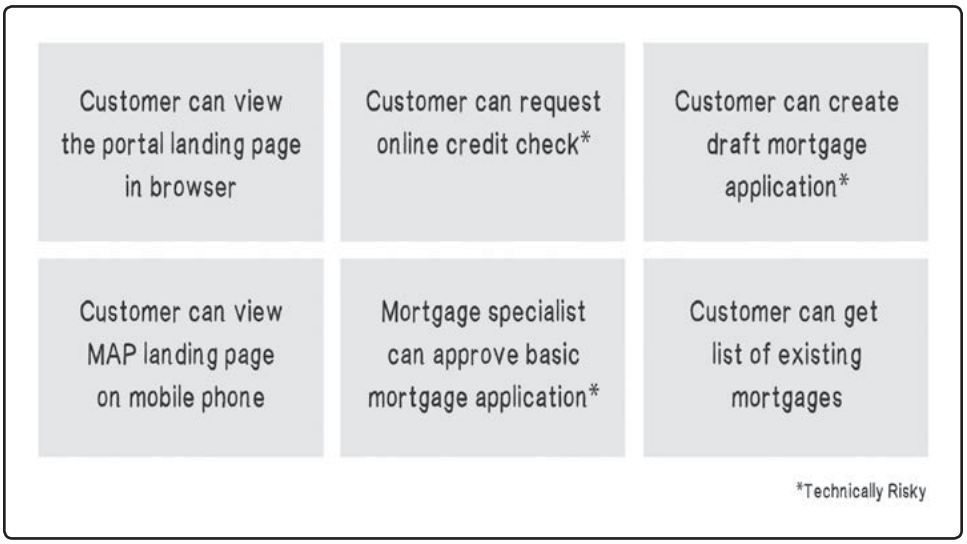

Customer can view the portal landing page in browser
Customer can request online credit check*
Customer can create draft mortgage application*
Customer can view MAP landing page on mobile phone
Mortgage specialist can approve basic mortgage application*
Customer can get list of existing mortgages
*Technically Risky

Figure 6.1 The stories for iteration 1.

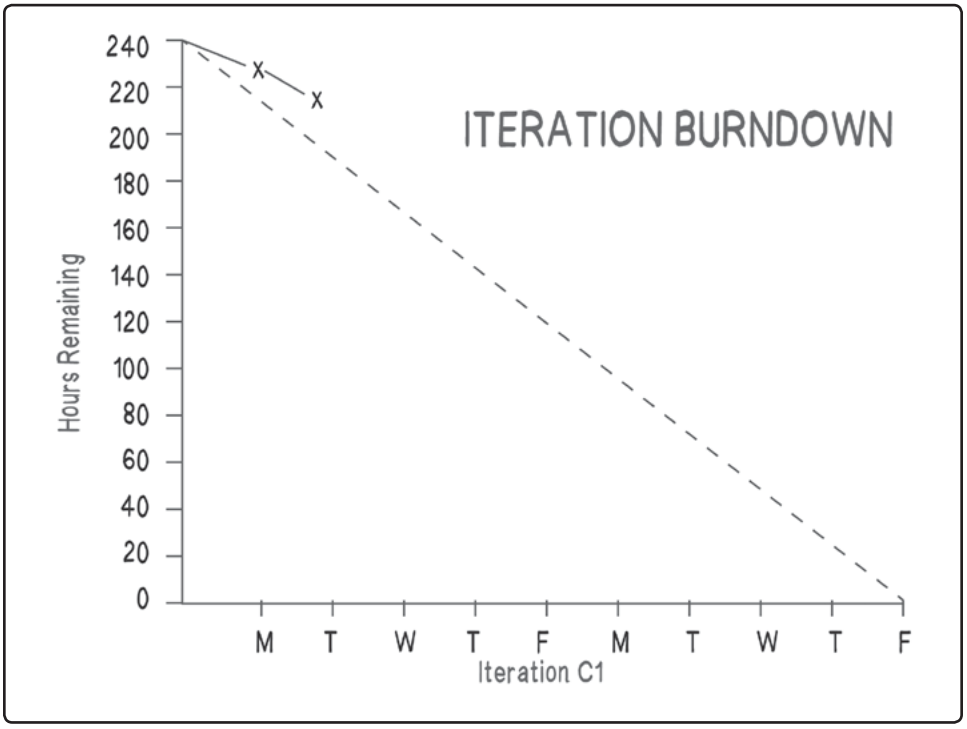


Figure 6.2 The MAP team's task board.

Terry facilitates the meeting, asking each team member to describe what they plan to do today, if anything is slowing them down or blocking their progress, and if they need help from anyone to complete their work.

After the coordination meeting, Terry creates an iteration burndown chart on a piece of flip chart paper and pins it next to the team's task board to help keep the team on track to complete all the work that they have committed to for this iteration. The burndown shows 240 hours of work remaining and will be updated every day as hours are reduced by work being completed.

During the day, Terry books an iteration review meeting with the stakeholders to be held the following Friday afternoon at the end of the iteration. He also books a meeting with the team to follow this review—known as a retrospective—to discuss the iteration results and how the team can improve.

Day 2. The team gathers around the task board for their daily coordination meeting at 9:00 a.m. sharp. Terry reminds the team that although yesterday's meeting ran for about 30 minutes, the goal moving forward is to keep the meeting to under 15 minutes. Terry also points out a new sheet that he has pinned next to the task board that describes his proposal for a definition of done (DoD). He explains that for each of the work items on the board, it is important that everyone understands what being "done" really means. The sheet looks like this:

Definition of done (DoD):

- All code tested and checked in
- All unit tests passing
- All acceptance tests written and accepted as passed
- Usability standards met
- Peer reviews completed
- Architecture handbook updated if necessary

This sheet will serve as a reminder that there is more to considering a work item being done than the code running and some basic testing being complete. The team discusses the proposed DoD and the implications for how they're going to work. During the discussion, Danny points out that Terry has forgotten to include the need to update the help screens to reflect changes to the solution. The DoD is updated accordingly, and it's expected that the DoD will evolve as the team learns more about the needs of their stakeholders.

As each team member gives their update for what they accomplished yesterday, what they plan to do today, and if they have any "blockers" to their work, Terry tracks how many hours the team has reduced on the task board. Once Ashok, Debbie, and Danny have given their updates, Terry provides his own update. Adding up the hours that the four of them have reduced off the tasks, he comes up with 15 hours in total. However, as the burndown chart of Figure 6.3 shows, in order for the team to reduce the total of 240 hours to zero by the end of the iteration, the team needs to burndown 24 hours per day. Terry brings this to the attention of the team and says, "We didn't get as much done yesterday as we had hoped, but I understand we are all new to this process. Additionally, I know that we were spending some time on activities not on our task board, such

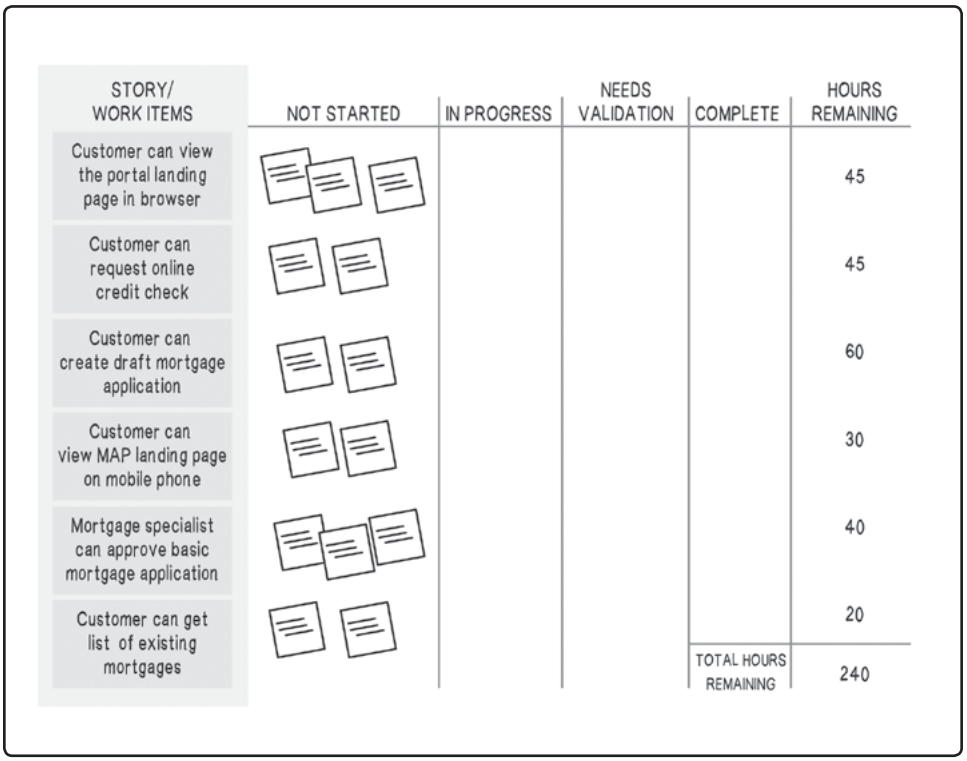


Figure 6.3 The iteration burndown chart.

as tuning our build scripts and other setup related to our development environments. So as the iteration plan shows, we have fallen nine hours behind. As a team, we'll need to figure out how to get back on track."

Day 3. During the daily coordination meeting, some issues surface. Danny says that he is struggling with his current task that he originally estimated would take three hours. He can't seem to figure out how to solve a technical issue, and he also thinks that his estimate was far too low. Not only did he make no progress, but he thinks that there are six hours still remaining. Terry explains that this is not unusual, and luckily, the team has some contingency in the iteration plan for unplanned work. Terry asks someone to volunteer to pair with Danny to get over this technical hurdle. Unfortunately, no one volunteers because they are most concerned about falling behind on their own work. Carlos, the coach, steps in and reminds the team that getting all the work done in order to meet their commitments to the product owner is everyone's responsibility. Debbie speaks up and volunteers to help Danny. As a result of increasing the estimate for Danny's task and not making progress, the team actually only burned down 10 hours instead of the expected 24. The burndown shows that the team is quickly falling behind.

While this is a definitely a problem, the good news is that using a chart like this exposes scheduling problems immediately and makes it transparent to the whole team. Danny volunteers to stay late tonight to try to get back on track. The rest of the team decides to stay and help out. Ashok can't stay as he needs to pick up his son from daycare, but promises to dial in on the VPN later this evening to work on his tasks.

Days 4-7. The team continues work and makes up for most of the time lost. However, it looks like they will not get all six stories completed by the end of the iteration.

Day 8. During the daily coordination meeting on the eighth day, the team discusses what to do about the fact that they will not meet their commitment of the planned work for this iteration. Carlos advises that it is better to get as many work items "done" according to the DoD that is posted, rather than 80% done on each story with nothing completely done. The team decides to stop work on *basic application approval*, which Danny just started working on yesterday afternoon, and will now focus on completing the other five work items. They notify Patricia, the product owner, immediately so that she can inform the stakeholders and thus retain team transparency.

Dick, the database administrator, is skeptical about working in an agile manner, even after the DAD training the team received. Throughout the iteration, he has paired with people whenever any database development was needed, such as creating a new table or adding columns to an existing table, but each time he has done so, he has complained about wanting to do the entire database design up front. During the coordination meeting today, he identified this as a blocker, and Carlos offered to discuss agile strategies for database development[1] with him after the meeting. This conversation doesn't go well, even though Carlos shared stories about previous experiences on other teams in similar circumstances. Dick decides that he knows best, and decides to secretly work on detailed logical and physical data models in his spare time so as to "save the team" when they discover that he is right after all.

Day 9. Ashok and Debbie pair together to implement the *save draft application* story, one of the three key stories that help to prove that the team's architecture strategy is sound. Early that week, Ashok completed work on *request online credit check*, also a technically critical story. The previous day, the team chose to drop *basic application approval* from the current iteration, the third technically critical story. As a result, the team will be unable to fully prove that their architecture works until at least Iteration C2. Terry works with Patricia to cancel the existing milestone review meeting (scheduled for tomorrow) and tentatively reschedules it into the same time slot at the end of Iteration C2. They also send out an email explaining why the review needs to be rescheduled.

Day 10. The team spends the morning stabilizing the work items/stories that they have completed, doing exploratory testing, and fixing last-minute defects.

The iteration review. In the afternoon, an iteration review is held with the stakeholders. DAD suggests that this review should be more than just a demonstration. It also makes sense to review the team's status against the vision that was agreed to at the end of the Inception phase. As such, Terry has prepared a simple presentation to cover the following topics:

- **Set the context.** Where are we in relationship to the release plan?
- **Iteration results.** Did the team deliver all the work items that they committed to?
- **Team performance.** What is the throughput of the team (velocity)? Is quality good?

[1] See AgileData.org.

- **Solution demonstration.** What value did the team actually produce?
- **Update on the vision.**
  - Are we on track? Based on the velocity of the team, can we deliver according to the dates in the release plan?
  - What is the status of the outstanding risks?
  - When will the architecture be proven?
  - Are there any issues that need to be addressed?
  - Q&A

The team isn't entirely confident that the new functionality will work as expected during the demo, since the application was periodically crashing during testing. Ashok insists on doing the demo so that Patricia won't break it. Carlos lets the team conduct the demo this way, but says that he wants to discuss the approach during the retrospective after the iteration review meeting. Sure enough, the demo crashes when the last new feature is being demonstrated. The stakeholders poke fun at the team and they feel a bit embarrassed. Nevertheless, the stakeholders are still impressed that they saw some working software only two weeks into the effort. They did say that they are a bit disappointed that the team didn't meet its commitment on all the functionality. Carlos mentions that this is a common pattern for new agile teams in that they tend to be overly optimistic during their estimating. Agile is new to the team and it will take a few iterations for them to learn to work effectively together and be certain that they are making commitments that they can keep.

The retrospective. After the meeting ends, the team stays behind to have their retrospective meeting. Carlos facilitates the meeting, asking each team member to describe what they thought went well, what didn't go well, and some ideas for how the team can improve its performance. They discuss the following issues:

- Terry feels that the team did a great job for a team new to agile doing their first iteration. Carlos agrees.
- Tara expresses concern that the burden of testing was on her shoulders and most of the stories were not available for proper testing until the seventh day of the iteration. She didn't have time to cover as many test cases as she had hoped. The team agrees to pitch in and help with the testing more, and to test one another's work, not just their own. They also decide that they need to try to build the stories in priority order and make them available to Tara and Patricia as early as possible in the iteration.
- Danny admits that he should have asked for help with his technical issues earlier. The team agrees that they should try more pairing when developing tricky parts of the solution.
- The team recognizes that they are still new to continuous integration (CI) and that they need to work on their approach a bit more.
- The team was only able to deliver 16 points, instead of the 20 points of work that they had intended. They decide that they should only commit to 16 points for the next iteration, and take on additional work if they finish this work with time to spare. Fortunately, with two contingency iterations, the team can still deliver all functionality according to the planned dates in the release plan.
- The team recognizes that they should have fleshed out the high-priority items on the backlog a bit better so as to have a smoother planning session. The team realizes that they will suffer from

the same problem the next day when they plan the next iteration. Patricia commits to doing some look-ahead analysis modeling for each iteration from now on, and Ashok similarly commits to leading team members in doing look-ahead design modeling. Carlos points out that this is sometimes called backlog refinement in Scrum, although DAD prefers clearer terminology such as look-ahead modeling.

As a result of the retrospective, the team develops a handwritten list of improvements that they captured on their whiteboard. Realizing that they likely can't implement all of the improvements immediately, they decide to prioritize them so that they can focus on implementing a few key improvements at a time. Displaying their list publicly as an information radiator helps to keep the improvement plan in everyone's mind.

Chapter 7

Construction Iteration C2

Iteration planning session. The team comes in Monday morning refreshed and ready to start a new iteration. Patricia walks the team through the highest priority work items that collectively add up to 16 points. For each story, she explains the functionality and the acceptance criteria. She has brought a few screen mockups to the planning session that she shares with the team. The mockups are a good start, but as they found during the first iteration, they aren't sufficient. For two of the stories, she finds that she needs to draw a flowchart to describe the business logic supporting the story. She also takes time to describe several business terms, often to respond to questions from Dick, who likes to understand these sorts of things. The other team members also find that these conversations have helped to clarify several concepts about which they were unsure.

For each story, Ashok leads the team in a discussion regarding the technical strategy that they will take to implement it. Based on this "just-in-time" design, the team identifies its development and testing tasks in a similar way to the last iteration. At several points, Danny and Ashok work through an idea on the whiteboard while others provide input. Although this modeling proves to be of value, the team realizes that it adds time to their planning session.

Since the team is now more familiar with the process, the planning session goes a bit smoother than it did in the previous iteration. However, it takes longer than expected due to the need to work through the business logic and implementation details. This further reinforces their decision to start doing look-ahead modeling each iteration, one of the learnings identified in their retrospective from the last iteration. Terry volunteers to clean up the task board from the previous iteration and set up the board with the new work items and tasks. After lunch, the team is ready to jump in and start working on the items

Days 1–5. The team starts doing the work required to implement the stories, including the story they didn't complete in Iteration C1 that they've picked for this two-week iteration. Although everyone is sitting together in the agile pod, they are still working (for the most part) separately. Whenever someone runs into difficultly, such as how to specifically write a section of code, they ask someone else for help. Danny, Debbie, and Terry typically ask Ashok for help, as he has many years of development experience. Every so often, a team member asks Patricia or Barbara to do a bit of just-in-time (JIT) model storming with them to work through the details of some business logic or how a user interface feature should behave.

Dick continues to pair with other team members on database functionality when needed. He also continues to work on the detailed data model and pretty much finishes it this week.

Tara and Carlos work on getting the team's continuous integration (CI) strategy up and running. Carlos has done this sort of thing before, and Tara is eager to learn more as she is intrigued about how automated regression testing fits in. Furthermore, Tara has little testing work to do this week while she waits for other team members to sufficiently complete their work before she can start hers. Although the others on the team are doing their best to write unit tests, it certainly isn't their

strong point. Tara feels that this isn't how the team should be approaching testing, and it doesn't reflect the test strategy that the team had developed during Inception, so she brings it up with Carlos. Carlos agrees and tells her that this is an issue he intended to bring up with the team once the CI infrastructure is in place. He also suggests that improvement suggestions might be better coming from her, and that she should think about it a bit.

The daily coordination meetings each morning are starting to run better, although they often last 20 or even 25 minutes, instead of the desired 10–15 minutes. Some people, particularly Danny, tend to show up late for the meeting, forcing people to wait. Both Ashok and Debbie, and sometimes Terry, go into too much detail and often try to start solving problems during the meeting instead of waiting until afterward. Danny suffers from the opposite problem in that his updates are often too short and neglect to include important information that other team members need to know. Carlos gently coaches the team every day on these issues.

On the fifth day, Carlos takes Danny aside and asks him why he's often late. Danny says he has trouble getting up some mornings, and besides that, he doesn't think that there's much value in the coordination meetings anyway. Carlos describes to him why it's important that everyone attends on time and is prepared. He asks Danny to do so, just like the rest of the team members. Danny agrees, but Carlos can tell that Danny still isn't fully convinced.

Day 6. Patricia and Barbara, the business analyst, spend two hours in the afternoon doing some look-ahead modeling with several stakeholders, including Berhard, to explore the next eight highest priority stories on the work item list. In the process of doing this, they create a collection of hand-drawn screen sketches, a flowchart overviewing the business process supported by the stories, and several business rule definitions that they capture as point-form text. After the modeling session, Patricia decides to schedule a recurring meeting in this time slot to perform look-ahead analysis.[1] Her intention is to invite the appropriate stakeholders two weeks in advance, as she realizes that the people she needs to work with will vary each time.

WHY ARE WE MODELING REQUIREMENTS THAT WE DON'T NEED TO IMPLEMENT YET?
A BIT OF THINKING NOW WILL SAVE A LOT OF TIME IN THE NEXT ITERATION.

Day 7. Debbie, with the help of Ashok, finishes work on the *basic application approval* story that had been dropped from the previous iteration. With this work accomplished, the team has done sufficient work to prove that their architecture strategy is viable via working code (sometimes called a "walking skeleton"). After a quick review by Terry to verify that they have met the acceptance criteria for *basic application approval*, he is confident that he can safely confirm the lightweight milestone review meeting that had been tentatively scheduled for day 10.

[1] For more about modeling strategies, see AgileModeling.com.

Day 10. As with the first iteration, the team spends most of the morning on hardening activities. Patricia prepares for the demo, practicing several scenarios with the working solution that will show the new functionality that was implemented this iteration. Ashok and Terry meet for half an hour to finalize their strategy for the milestone review meeting in the afternoon.

The demo. The demo goes much better this iteration for two reasons: First, the team completed all of the work that they had committed to at the beginning of the iteration, which pleased Berhard. In this iteration, the team delivered a total of 19 points, which is an improvement over the 16 delivered in iteration C1. Carlos explained that it is common to see a team's velocity increase during the first few iterations as the team becomes more efficient and implements its own process improvements from their retrospectives. Second, Patricia was much more comfortable running the demo this time. Patricia invited two business stakeholders to the demo, Connie and Rod, who had been involved with initial requirements modeling, but who hadn't yet seen what the team had produced to date. During the demo, they are excited about what they see, but are perplexed as to why there is so little to see. Although Patricia and Berhard have explained the new agile approach to them, they didn't realize at first that they would only see a partially complete solution. Once they get over this misunderstanding, they begin to suggest new ideas for what could be built. Patricia tells them about the next modeling session in a week and a half, and invites them to be involved. Connie and Rod promise to make room in their schedules and, more importantly, to think some more about what they really want the solution to do.

The Proven Architecture milestone review. Ashok walks Enrico, the enterprise architect; Oliver, the operations manager; and Chris, the CSM, through the working solution. The rest of the team, including Patricia, also attend this review to be available to answer any questions that may arise, and hopefully to pick up some insights into how architects think. Ashok's focus is on working through three stories: *save draft application*, *request online credit check*, and *basic application approval*. These stories represent a beginning-to-end walking skeleton of the solution that proves the architecture works. Ashok explains these stories to the stakeholders, describing how the implementation of the stories stresses the architecture in a meaningful way. He does this by sketching the architecture solution on the whiteboard as he describes each story. Enrico asks a few pointed questions regarding how Ashok's strategy fits into the company's overall technology roadmap. Ashok replies that it's the team's architectural strategy, not just his, and then goes on to describe how, during Inception, the team's architecture modeling was influenced by the roadmap. Patricia points out that the scoping effort was similarly influenced by the business roadmap.

Toward the end of the review, Chris questions why the *request online credit check* story was implemented this early in Construction. Patricia jumps in, saying that the story was, in fact, a medium-priority requirement originally, but that Ashok convinced her to move it to the top of the work item list so as to prove the architecture. Chris says that's exactly the issue that he's bringing up, saying that Scrum teams would work in business-priority value. Ashok replies that an advantage of Disciplined Agile Delivery is that it focuses on risk reduction early in the life cycle, including both business and technical risk. Enrico and Oliver both agree that this strategy is much better than what they've seen with Scrum teams in the past, saying that they have a much better understanding of what the team is actually doing and are impressed that they not only thought through important architectural issues, but can actually show that they work. Enrico observes that this has worked better than the onerous architecture reviews that many of BigBank's traditional development teams are still doing.

The retrospective. Terry leads the retrospective, starting it by asking everyone to discuss what they feel is going well. Overall, people are happy with how things are going. They like the more collaborative

style of working, and everyone has an anecdote about how they learned something from someone else through working closely with them. Everyone is relieved that both the iteration demo and the milestone review went well, although there is a bit of concern about how the requirements may expand, given the exuberance of the stakeholders during the demo. The team also reviews their improvement list from the previous iteration and discusses where they've made some headway.

When Terry asks people where they need to improve, it becomes clear that quality is the main issue. Carlos points out that the need to spend so much time to harden at the end of the iteration is an indication that the team isn't testing sufficiently, or fixing any problems early in the iteration. Danny says that sometimes he needs to code for a few days before he has anything suitable for Tara to test. Carlos points out that the team is currently following a strategy called "testless programming," where developers write a bunch of code and then ask someone else to test it for them. Debbie claims that she and Danny are doing some unit testing as they code, which Carlos says is a good start, but that it's possible for the developers to do more testing themselves and thereby reduce the overall cost and time to fix any problems with their code. Carlos walks the team through the goal diagram for Produce a Potentially Consumable Solution, which is shown in Figure 7.1.

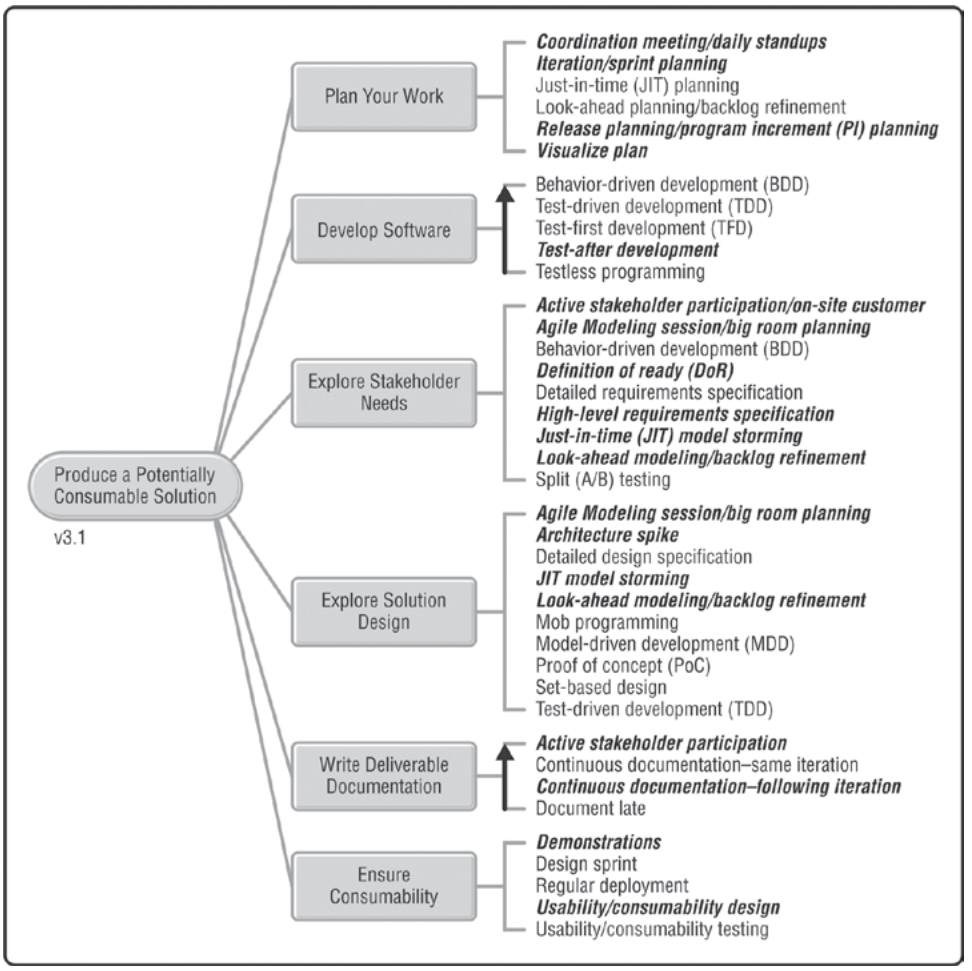


Figure 7.1 The goal diagram for Produce a Potentially Consumable Solution.

As you can see, DAD supports two other, more effective approaches to programming: test-after programming, where people write a little bit of code and then test it thoroughly (ideally, writing automated regression tests when doing so), and test-first (or test-driven) programming, where you write a single test and then just enough production code to fulfill that test. After a bit of discussion about the advantages and disadvantages of each approach, the team decides to try a test-after approach going forward for now. Although a test-first approach would be preferable, the team feels that they don't yet have the discipline to make it work. Carlos points out that once the team gets good at test-after programming, they should then consider taking the next improvement step to a test-first strategy.

Tara says that she's happy to hear this. She has been concerned that the team is flagging a bit to implement the testing strategy that they had agreed to during Inception. To help move things along, Tara offers to be available to pair with people while they are writing code to help them to work in a test-after manner. Carlos brings up how the team's continuous integration (CI) strategy is operational, but the team isn't yet taking full advantage of it. Everyone agrees, and Carlos commits to working with everyone next week to give them some one-on-one coaching in CI.

Danny asks whether the team should take on more points of work in the next iteration, since in this iteration they completed 19 points versus their estimated velocity of 16. Carlos recommends tracking their velocity trend for one or two more iterations before changing the expected velocity and adjusting the release plan forecast.

The team finishes the day by updating their "management artifacts." The risk list is updated to show that the architecture has been proven and accepted by the appropriate stakeholders. The team also adds the risk that the requirements may expand given the positive feedback received during the demo.

Chapter 8

Construction Iteration C3

Construction Iteration C3 begins on an uptick. The team is excited because the second iteration was successful and the general consensus is that it will be the same for this iteration. Terry leads the team through iteration planning, which takes a bit less than three hours. This is much faster than what has happened in the past. This is due, in part, from the team's greater experience with iteration planning, but mostly due to the look-ahead modeling efforts led by Patricia and Ashok.

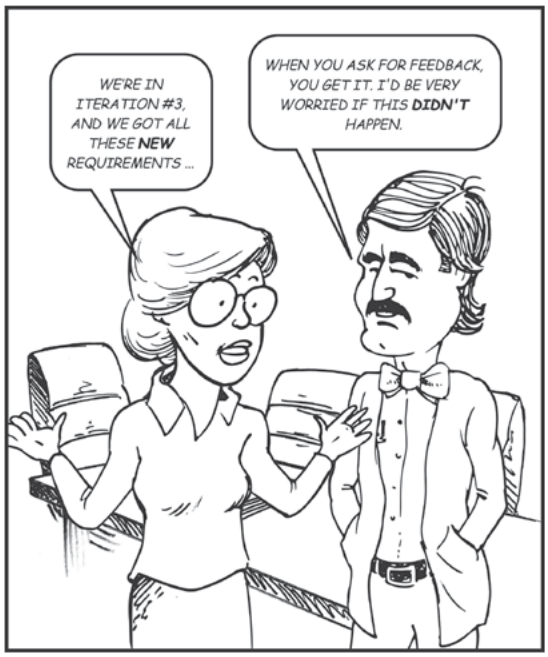
The daily coordination meetings are running much better. Everyone is showing up on time and, for the most part, staying focused on coordinating their day instead of trying to solve problems in the meeting. However, Danny is still struggling to participate effectively in the meetings.

Day 2. During the team coordination meeting, both Danny and Debbie ask Tara if she has some time to pair with them separately to help them test their work. To do this, Tara sits right beside a developer and works with them to write both production code and the unit tests to validate that new code. After a bit of time to get used to it, Tara and the developer swap the keyboard regularly, talk through the logic of how to test something, and most importantly, learn new skills and ways of thinking from each other. Throughout the rest of the iteration, Tara spends about two-thirds of her time pairing with either Danny, Debbie, Ashok, or Terry in this manner. The other third of her time is spent on other forms of testing, including functional, integration, and stress testing. During the iteration planning session the day before, Tara indicated to the team that she had intended to spend most of her time pairing with others to help them test. This is in line with the test strategy that the team had agreed to earlier during Inception.

Day 6. Early in the afternoon, Patricia leads a look-ahead modeling session with several stakeholders to explore upcoming requirements. During this session, the stakeholders not only describe how the stories for the upcoming iteration should work, they also identify several new stories and drop a few of the low-priority requirements. These changes result in what looks to be a 20% increase in scope.

Patricia goes to Carlos in a bit of a panic, asking for help to deal with this "scope creep." Having worked with IT teams before, albeit ones following a traditional approach, Patricia is worried that this increase in scope will put the team at risk. Carlos points out that a big spike in new requirements early in Construction is quite common on agile teams. It is the natural result of greater stakeholder participation—when you produce a consumable solution on a regular basis, show it to people, and ask for their feedback, you typically get it. It's healthy for the requirements to evolve as it increases the chance that the team will build something the stakeholders actually want as opposed to something built to specifications.

Day 7. During the daily coordination meeting, Patricia informs the team about the new requirements. Terry suggests that the team spends some time today to have Patricia walk through the new requirements so that the team can identify the potential impact. The team decides to do this immediately following lunch. Patricia begins the session describing the changes and the reasons behind them. After a bit of discussion around how to implement the new stories, the team


WE'RE IN ITERATION #3, AND WE GOT ALL THESE NEW REQUIREMENTS ...
WHEN YOU ASK FOR FEEDBACK, YOU GET IT. I'D BE VERY WORRIED IF THIS DIDN'T HAPPEN.

is convinced that the new requirements will not impact their architecture strategy at all. They decide that this is due to the initial modeling they did during the Inception phase.

Terry leads the team through sizing the new requirements using the relative-mass sizing technique, taking about 30 minutes to do so. The total time for the impact assessment meeting was a bit more than 1.5 hours. After this, everyone on the team, except for Terry and Patricia, goes back to work. Terry and Patricia work together to update the release burnup chart, which now shows that the team is likely to need 12 Construction iterations in total. Because Berhard, the sponsor, has made it clear that the schedule is firm, Patricia realizes that she needs to call a meeting to rethink which requirements are "must haves" for this release and which can wait for a future release.

Day 8. During the coordination meeting, Terry and Danny discover that they both worked on the same functionality the previous morning. Although this was a bit frustrating for both of them, for Danny it was a valuable lesson in why everyone needs to actively participate in team coordination meetings.

Day 9. Patricia facilitates a one-hour meeting with Berhard and the stakeholders who were involved with the modeling session on day 6. During the meeting, she negotiates with them to allocate some of the lower priority requirements to be implemented in the next release. Fortunately, management has learned that treating this initiative as a series of releases, similar to a typical product management strategy, is a better approach than funding one project at a time. Knowing that there will be more funding allocated to a future release makes these scope negotiations much easier. About 70% of the requirements, including most of the new ones, are marked as "must haves" for the first release. All others are marked as "nice to have" that ideally will be implemented in the first release, although if they get pushed to a future release, the stakeholders will accept it. This categorization will enable the team to make their delivery date. Terry also attends the meeting to represent the team, but for the most part spends his time listening to the discussion.

Day 10. The team finds that it has much less hardening to perform during this iteration. This is due to doing more testing earlier in the life cycle. The demo goes very well. The stakeholders like what they're seeing and feel very optimistic about the progress being made. This sense of optimism is new to them, as their previous experiences working with IT have always been problematic. Due to the degree of their active participation with the team and through regular demonstrations, they realize that they are less likely to encounter nasty surprises late in the life cycle, which has been typical in the past.

Terry shares the updated release burnup chart with the team, shown in Figure 8.1. The chart now shows two requirement lines, one showing the number of "must have" points and the other showing the grand total of points. Patricia describes to the team what happened during the prioritization session with the stakeholders the day before. Terry then explains that, with the new version of

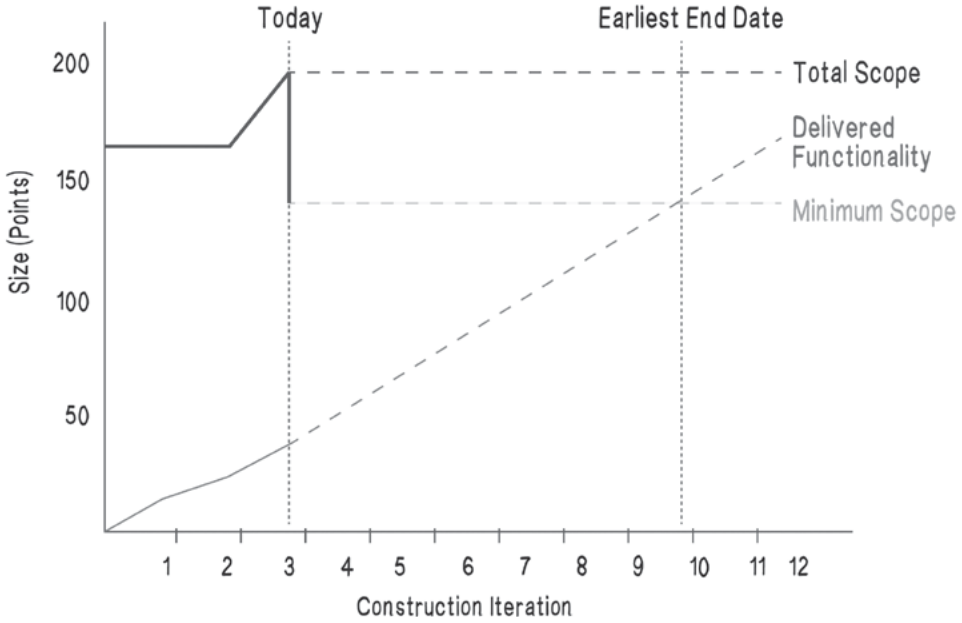


Figure 8.1 The MAP team's release burnup chart.

the burnup chart, the point where their burnup line meets the must-have line is the earliest time that they can decide to release/transition their work into production. At this point, they will be completely done all of the requested work. This is known as the Sufficient Functionality milestone.

The retrospective. The team begins with a review of how they're doing with implementing the process improvements from previous iterations. Their approach to testing has clearly improved, although they have much more work to do. The iteration planning session went smoother this time, and everyone hopes it will be even better in the next iteration. Team coordination meetings are also going better each day as well, although they sometimes last up to 20 minutes, so there is still room for improvement.

THE SOFTWARE IS POTENTIALLY SHIPPABLE!
OH YEAH ...
IS THE DOCUMENTATION UP TO DATE? WHAT ABOUT HARDWARE CHANGES? BUSINESS PROCESS CHANGES?

Ashok says that he's a bit worried about the level of documentation that the team is producing. A few days earlier, Oliver, the manager of the operations team, asked Ashok whether the team had started to work on the system overview documentation that's required as part of the overall solution. Ashok said that the team had been capturing notes and photos of diagrams in their team wiki, but had not yet started developing any form of "deliverable documentation." Although they're still early in Construction, Ashok feels that they really should be evolving the supporting documentation along

with the software that they're building. This reflects DAD's philosophy of developing consumable solutions, not just working software. Carlos suggests that the team adopts the practice of continuous documentation, where they write the supporting documentation in parallel to writing the code. He suggests that this can be done in the wiki, and that all the team needs to do is invest a bit more effort to capture sufficient information there. Carlos also offers to pair with anyone needing help.

On a related note, Dick reveals that he's been maintaining a detailed data model so far. Danny jokes that this was a well-known secret, because everyone on the team knew this was going on. Dick admits that he thinks he's wasted a lot of time maintaining this detailed model. The requirement spike earlier in the week caused him to do a couple of hours of rework on his model, some of which was removing data structures that were no longer needed due to dropped functionality. He's come to realize that evolving the database schema via database refactoring techniques is very safe and straightforward, removing the need for up-front detailed data modeling. Instead, he's going to perform database design work on a just-in-time (JIT) basis from now on.

Chapter 9

Construction Iteration C7

It is the seventh Construction iteration, and the team is working smoothly. During iterations four through six, the team successfully evolved the mortgage application portal (MAP), adding new value to it every two weeks. They consistently worked together to learn from each other and improve their approach based on the ideas they generated during each retrospective. There were a few bumps along the way; in particular, there were a few heated discussions between team members when disagreements arose as to how to implement certain features. Luckily, Ashok led the team through these hard technical decisions, exactly as an architecture owner should.

Stakeholder collaboration. The stakeholders are very happy with the progress that is being made, and are regularly giving feedback to the team during both scheduled and impromptu modeling sessions, as well as demos. Patricia is comfortable in her position as product owner, and is finding the role both rewarding and challenging. It can be hard, at times, for her to get access to the right stakeholders, as they have their "real jobs" to do. And, of course, the stakeholders all have differing opinions and goals, making it difficult for her, at times, to negotiate requirement priorities with them. Requirements are still evolving, with new ones being added all the time, which is a natural consequence of regular stakeholder participation.

Patricia and Berhard begin working with Mindy's team to develop a marketing strategy for MAP. This is the first time that Mindy's team has had to develop a strategy for a customer-facing solution that will be regularly released in small increments. Mindy is very excited about working in an agile manner, because it feels natural to her. Mindy asks Patricia to arrange some training for her staff to ensure that they understand how agile works and what they need to do to interact effectively with the development team. Patricia asks Terry and Carlos to run a two-hour training session for Mindy's marketing team, and both Patricia and Mindy attend the training to help underscore the importance of this new approach to BigBank. Mindy's team will work closely with Patricia and Terry from now on to ensure that the marketing strategy is synchronized with what the development team is doing. The plan is to begin the marketing campaign a few days after the system is released into production.

Test often, test early. Danny, Debbie, and Terry have all increased their testing skills through pairing with Tara. Lately, Tara has been working with Barbara, the business analyst, to automate acceptance tests. Progress was slow at first, because Barbara has always captured detailed requirements in documents, but she is now starting to see the value in capturing these details in the form of executable tests. Tara also works with Dick to create and maintain test data.

On the technical side of things, the team's continuous integration (CI) strategy is working very well for them. During Iteration C5, the team's test suite was too slow due to the growing number of regression tests. The team's solution was to create a test suite that would run in five minutes or less on their work stations. This test suite focused on the functionality developed during the previous iteration, as well as new functionality this iteration. To help speed up the tests that run on people's workstations, Dick introduced database mocks during Iteration C6. A second test suite is run in their team integration sandbox, which takes about 30 minutes in total. This test suite runs every time

someone promotes new code into that environment. There is a third test suite that runs once a night. This test suite takes several hours, and it includes stress tests and end-to-end integration tests.

Automated deployment scripts. The team has developed a script to automatically deploy their working build into a demo environment. This script runs when the nightly build is successful. The script removes the system from the demo environment, including the database, and then rebuilds everything from scratch in the demo environment. The team is thinking of reworking this script for use in deploying into production, an important step toward the DevOps strategy called continuous deployment.

As far as the database goes, Dick evolves the database schema as needed, and lately has been actively pairing with others so that they can learn fundamental database skills from him. His comprehensive data model is still being updated by him, but he decides to give up on all the up-front modeling and removes details that are not yet implemented.

Automated dashboard. During Iteration C4, the team begins to take advantage of the dashboard functionality provided by their development tools. As they do their work, the development tools are recording key events, such as whether a build was successful, when the build ran, how many tests ran, and so on. This web-based team dashboard shows important metrics, often in the form of graphs, generated from this event data. The team started using the dashboard in their own browsers and, at the suggestion of Carlos, Terry set up the dashboard to display on the team's monitor on the wall. This strategy is something that DAD refers to as development intelligence, an application of business intelligence (BI) technologies for software delivery teams.

Planning. The team has firmly committed to a delivery date based on 10 Construction iterations. During the previous iteration, it looked as if they would reach the point of having delivered sufficient functionality by the end of Iteration C9, but Patricia and Terry agreed that they would rather have that tenth iteration as buffer space. If they end up with spare time, they will simply implement a bit more functionality than the bare minimum required.

The team is also getting serious about deployment planning. They are working very closely with Oliver, the operations manager, and Samira, the support manager, to ensure that their teams are ready to accept the new mortgage application portal (MAP) being developed by the team. They have negotiated the release window that they are aiming for with Oliver—at BigBank you can only deploy into production on Saturday or Sunday between 3:00–5:00 a.m. They have also started planning how they are going to train Samira's help desk people. Samira's group has a training team that focuses on exactly this type of work. Terry offers to work with this training team to ensure that the training materials will be developed in time.

The team has kept the deliverable documentation up to date since Iteration C4 via a continuous documentation strategy. This documentation includes a system overview, online help pages for end users, an online support guide, and technical release notes.

The retrospective. Terry tells the team that Padma has asked him to stop updating the Gantt chart. She says that the information she is interested in is captured well on the release burnup chart, but not on the Gantt chart. This is a pleasant surprise for Terry, as he didn't expect that Padma would be receptive to agile reporting strategies.

Due to the improvements made in the way that the team approaches quality, there has been very little hardening effort at the end of the last two iterations. Debbie asks if there's a way that the team can use this newly available time effectively. Carlos brings up the idea of doing iteration planning for the next iteration on the last day of the previous one, effectively putting all the management overhead activities on one day. This is something the team decides to try for the rest of the Construction iterations.

Chapter 10

Construction Iteration C10

This is the last Construction iteration before Transition, and everything is running smoothly within the team.

The focus of this iteration is to implement the high-priority stories that were identified during the previous iteration, and to harden the solution. During Iteration C9, Tara shifted her focus from helping the team with unit testing and automated acceptance testing to testing in BigBank's preproduction environment. Tara wanted access to this environment much earlier, but BigBank has limited resources for this sort of testing. By working with the preproduction test team, she found several system integration challenges that her team wasn't able to simulate up until now. Tara also invests time to thoroughly test the deployment scripts within the preproduction testing environment.

The team chooses to capture these defects as new work items, which are prioritized and put on the work item list. Each defect is assigned a size of zero, so that they are not counted toward the team's velocity. The decision was based on the recognition that, from the point of view of the stakeholders, the team should have found and addressed these problems before claiming that the original work was "done." The team finds this a bit frustrating, but realizes that this is a learning experience for them. Carlos points out that having to spend an iteration or more hardening a solution is a common anti-pattern for teams new to agile.

Terry asks Tara to track the defects, by severity, in a simple spreadsheet. She tracks the severity of the issue, the date it was identified, a brief description, and the date it was resolved. This information is then summarized in a defect trend chart. Tara has done this sort of thing before using more sophisticated defect-tracking tools, but finds that the spreadsheet approach is sufficient for their needs right now. The defect trend chart is reviewed as part of the team coordination meeting each morning.

Day 3. During the morning, Terry reviews the deployment plan with Oliver and Samira. This plan describes how the team will work with Oliver's operations team to deploy the solution into production. The plan also indicates how the support team, managed by Samira, will be trained in time for the deployment. This training needs to occur over several days, because only a few help desk staff can be trained at any given time (the rest need to continue providing support to BigBank customers and staff members).

That afternoon, Terry and Patricia meet with Mindy, the vice president of marketing, and Mary Jane, who works for Mindy, to finalize the overall release schedule with her. The four have been meeting for 30 minutes on a weekly basis to coordinate the activities between the two teams. Mary Jane has been attending the iteration demos and has been actively working with Patricia to ensure that the marketing message reflects what the team is building.

Day 6. Patricia cancels her regular look-ahead modeling session, and instead spends the time crafting an email with Terry to communicate the release date to the stakeholder community.

Although key stakeholders have known about the target release date for a while, the wider stakeholder community only knew that it was coming soon. This email indicates the intended release date, a description of what MAP does, an overview of why this is important to BigBank, and an indication of when to expect further communications. This email is submitted to Berhard, the business sponsor, and Victoria, the vice president of IT, for their review. The following day, they jointly send it out to their constituents.

Day 10. Patricia decides to run a two-hour demonstration of the overall solution to the stakeholders. Where previous demos have focused on the new functionality implemented during that iteration, this demo walks people through critical, end-to-end functionality of the entire solution. Patricia decides to invite everyone who has attended a demo in the past, and most are able to attend. The goal of this demo is to both communicate what the team has accomplished and to verify that they have, in fact, produced sufficient functionality to justify releasing the solution into production. During the demo, several stakeholders indicate that they would like more functionality than what is currently implemented, and Patricia asks them whether they can wait until a future release for that functionality. Two of the stakeholders are unhappy with that idea because they have heard this promise before from IT, only to find out that the next release is years away. Berhard states that the plan is to get just enough functionality into production now so as to get to market quickly and that they fully intend to start into the next release once this one is successfully running in production. Victoria confirms that she will be keeping the team together to work on the next release. The two stakeholders are relieved to hear this and not only acquiesce to waiting until the next iteration for their desired functionality, but also offer to be involved with any requirements elicitation sessions. The demo is a success, and everyone is excited to hear that the deployment into production is planned to occur two weeks from now.

The retrospective. The team is very excited to be this close to releasing their solution in production. Several people on the team believe that the Transition phase is going to go very smoothly, given the quality of the work that they've done.

Terry shares the updated release burnup chart with the team. Terry compliments the team for both hitting the desired date and for implementing all of the "must have" functionality identified for this release by the stakeholders. He says there is clearly more to do in the next iteration, and fully expects that the team will receive more requirements given the positive response in the demo earlier today.

Both Danny and Ashok point out how they're looking forward to a bit of a break, as they're a bit worn out from having to consistently deliver a consumable solution every two weeks. Carlos agrees, pointing out that it's common for agile teams to slowly burn themselves out if they don't pay attention to working at a sustainable pace throughout Construction. Carlos thinks that this is something the team should work toward during the next release, and the rest of the team agrees.

Tara brings up the problem she ran into with preproduction integration testing. In hindsight, she feels that it was left until too late in the life cycle, although she doesn't see how it could have been done earlier, given her lack of access to the appropriate environment. Carlos describes a Disciplined Agile practice called parallel independent testing, overviewed in Figure 10.1, where the development team tests to the best of its ability, but also provides their working build on a regular basis to an independent test team. This team takes their build, and the builds from other

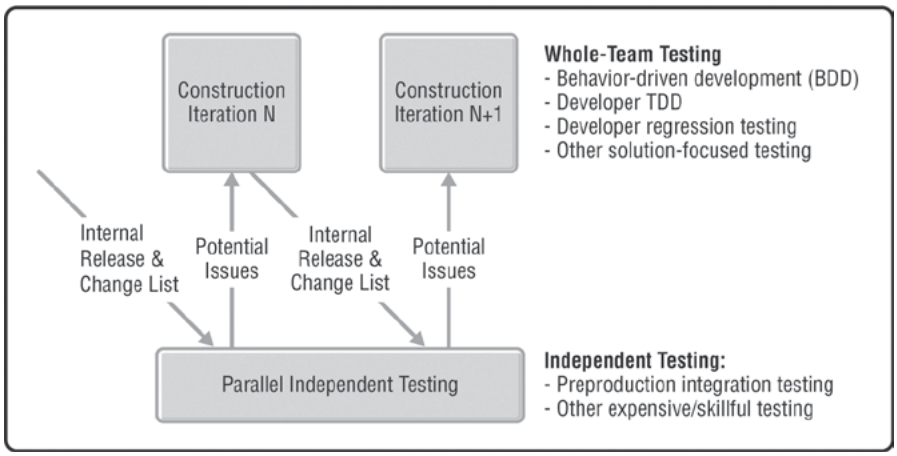


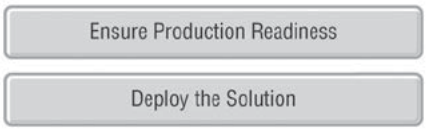
Figure 10.1 Parallel independent testing.

teams developing other solutions, and integrates them into a preproduction testing environment (sometimes called a "QA environment" in traditional organizations). The test team then performs the kind of testing that the development team isn't likely to be able to perform, and reports any potential issues back to the appropriate development team(s). Carlos believes that this will help the team to avoid the "hardening sprint" anti-pattern in future releases.

The Sufficient Functionality milestone review. Terry and Patricia meet with Padma, the PMO lead; Berhard; and Victoria to hold a milestone review. Terry wants to keep this as lightweight as possible. Terry summarizes the current status of the team, walking everyone through the burnup chart and the defect trend chart. Patricia and Berhard summarize the results of the demo earlier in the day for Padma, who hasn't been able to attend. After a brief discussion, everyone agrees that the Sufficient Functionality milestone has been passed successfully.

Chapter 11

Transition



The focus of the Transition phase in Disciplined Agile Delivery is for a team to release their consumable solution into production successfully. This should be done in as lightweight a manner as possible. In fact, when a team is allowed to stay together, this "phase" tends to evolve into an activity through automation—something we'll see in the next chapter. As you can see, there are two process goals to address during Transition: the first to ensure that you're ready to deploy, and the second to actually deploy.

Ensuring that the solution is ready to deploy. To do this, the team needs to:

- **Perform final testing and fixing.** The final testing effort is very short, at least by BigBank standards, due to the number of automated regression tests and the hardening work during Iteration C10. Tara, Danny, and Dick spend a lot of time testing in the preproduction environment, writing up a brief defect report for each potential issue they found. The issues are prioritized by Patricia as severity one through four. Any severity one defects have to be fixed, severity two defects should be fixed, and threes and fours are left to a future release if need be. These defects are tracked in Tara's spreadsheet and the defect trend chart presented in the daily coordination meeting. It is common for Berhard and someone from Oliver's operations team to attend these coordination meetings.
- **Support training of the help desk staff.** Training is delivered to Samira's help desk team over a period of six days. Each training session took three hours to run, and was presented to a portion of the help desk staff each time. During each training session, either Ashok or Terry attends so as to answer any specific questions participants have about MAP.
- **Finalize the deliverable documentation.** This proves to be a very minor effort due to the continuous documentation work performed by the team during Construction. Final reviews of the documentation are held with key stakeholders of each document, and appropriate changes are made. For the most part, the reviewers find minor consistency problems with the documents, a common problem when documentation is evolved over time (people sometimes forget to go back and update previously written material).
- **Support development of end-user training videos.** The team is asked to help record seven two-minute training videos to be delivered as part of the overall solution. There is a brief overview of MAP, and there are six "how to" instructional videos. Debbie, Patricia, and Barbara work together with Mary Jane to finalize the scripts. Mary Jane hires a professional actor to record the videos.
- **Validate the deployment scripts.** Ashok and Tara work with Oswald, an operations engineer from Oliver's team, to review and validate the deployment scripts. Tara has already thoroughly tested these scripts in the preproduction environment; in fact, she has been using them for several weeks successfully. The scripts are written so that environment information, such as the names and IP addresses of machines, is determined dynamically at run time. As a result, the review just needs to focus on the contents of the configuration file describing the production environment.

Deploying the solution. Oswald runs the deployment scripts to release the solution into production. Due to regulatory compliance needs, BigBank has a strict policy of separating the development roles and release management roles. Someone who is actively involved with development is not allowed to deploy into production—someone from Oliver's team must perform this work. This is why Oswald, and not a team member, is responsible for running the deployment scripts. It takes less than 10 minutes to do so.

Preparing for the next release. During some free time, the team meets for one hour to do release planning for the next release. This release took about six calendar months—three weeks for Inception, 20 for Construction, and two for Transition—but at the prodding of Berhard, the team decides that the next release should take only four months. Patricia holds some requirements-envisioning sessions to identify requirements for the following release. This comprises a two-hour requirements-envisioning session with several key stakeholders, followed the next day by a one-hour prioritization meeting. This effectively gives the team a head start on Inception for the second release. Unfortunately, the team doesn't have time to size the new requirements, so that work moves into a two-week "Inception phase" for release 2.

When you keep a team together so that they can work on the next release, you find that the Inception effort becomes much less (the team is already together, the environments are set up, the architecture is in place, funding is usually in place, and so on). In fact, most of the remaining Inception effort focuses on scoping and planning the next release, and that's typically a short effort. In this team's case, they are able to complete it in their spare time during Transition.

Although the average agile team spends about a month performing Transition activities, the mortgage application portal (MAP) team is able to do it in only two weeks. They are able to do this because of their focus on quality and testing issues throughout Construction, their adoption of the continuous documentation practice, their practice of automatically deploying their working build into demo and test environments (a big step toward continuous deployment), and working closely with operations and support staff to plan their deployment. Furthermore, the team feels that it could have been one week, had it not been for the need to spread out the training of the support staff. In the next release, they hope Transition will be faster, as the MAP will not be a new system and they will require less training.

Chapter 12

The Road to Disciplined DevOps

Over the long term, the mortgage application portal (MAP) application proves to be a successful system for BigBank. The first release, which focuses on providing mortgage information for existing customers and information about mortgage options, is well accepted by BigBank customers. It results in a measurable decrease in support desk calls and information requests in branches for mortgage information, providing an important cost savings to BigBank. The second release, which adds the ability for new and existing customers to begin the mortgage application process online, provides both a significant cost savings, as well as a revenue increase through a greater renewal rate. Subsequent releases add new and more sophisticated offerings.

In this chapter, we overview both the team's and BigBank's process-improvement journey. Over the span of several releases, the team evolves from their initial Agile life cycle to the Lean life cycle, and then eventually to the Continuous Delivery: Lean life cycle. Given the success of the MAP deal with DAD, BigBank decides to try out DAD on several other development teams. They realize that the enterprise aspects already built into DAD would have taken them years to figure out on their own, had they stayed with Scrum. The team's process improvements, particularly around evolving into the Continuous Delivery: Lean life cycle, dovetail nicely into BigBank's overall Disciplined DevOps strategy.

Release 2

Release two begins with a two-week Inception phase. During this period, the team spends several hours sizing the new requirements that Patricia has identified the previous week. Debbie and Ashok each decide to take the first week off for vacation, and Dick takes off both Fridays so that he can have two long weekends. The team also takes a three-day introductory course in test-driven development (TDD) the second week, as this is a practice that has intrigued several team members ever since they started regression testing within the team during release one. This phase also overlaps with the warranty period for release one, where the team had to be available to fix any production problems. During the first week of Inception, a few severity two and severity three issues are identified and fixed. A patch is deployed into production, once again working with Oswald over the weekend.

The Construction phase consists of seven two-week iterations. Here are the highlights of what happens:

- **TDD.** The team's adoption of TDD is a bit rough at first, but goes well over time. Carlos spends a lot of time coaching team members in TDD, pairing with each team member for an hour a day for the first three iterations. Tara also spends similar amounts of time pairing with people to help them gain basic unit testing skills. By the end of the release, everyone is comfortable with taking a test-driven approach, although they feel that they still need to work on growing their skills. Team members with good design skills to start with find it much easier to pick up TDD than those without such skills.

- **Remote work.** Before agile was introduced into BigBank, it was common for people to work one or two days a week from home. The scrum coaches that BigBank hired insisted on colocated teams, which from the point of view of reducing overall risk is a great idea. However, from the point of view of the people who are used to working from home, sometimes it is a great inconvenience. During Iteration C2, Carlos, the DAD coach, suggests that now that the team has gelled, they can safely experiment with letting people work from home occasionally. After discussing it with the team, Danny and Ashok begin to work from home one day a week. During Iteration C4, Debbie also starts to work at home one day a week. To support dispersed team members, two tools are adopted, one for agile management and the other for chat. The agile management tool enables the team to manage their work item list and task board, and will generate critical management reports, including the iteration burndown and the release burnup charts. The chat tool supports both text-based chat and videoconferencing, a feature that team members frequently use for collaborating and joining the daily coordination meeting whenever someone is working from home.
- **Improvement tracking.** During Iteration C1's retrospective, Carlos suggests that the team begins to measure their improvement efforts. He describes a DAD technique called measured improvement, where at each retrospective the team rates how well they are doing at addressing the previously identified improvements. The way that they do this is that they go through the list one at a time, and each person is asked to rate, on a scale of 1–10, how well they think the team is doing. The votes are captured in a spreadsheet so that the trends can be publicly displayed in the team room as one of the information radiators. If an improvement starts to trend downward, that's an indication that the team needs to refocus on that improvement. When an improvement levels off for many iterations, that's an indication that the team has either plateaued and may need to consider another strategy, or that it has adopted the improvement as best it is going to and doesn't need to focus on it any longer.
- **Guided continuous improvement (GCI).** Carlos also introduces the team to the concept of GCI,[1] where the team leverages the DA tool kit to help them increase their rate of process improvement. When the team identifies issues that need to be addressed within a retrospective, Carlos shares his advice, but more importantly, guides the team in using the book *Choose Your WoW!* to identify potential improvements to experiment with.
- **The team evolves.** The team interviews several potential candidates to join the team, and brings on David, a developer who is new to BigBank, during the second week of Iteration C3. At the start of Iteration C5, Doug, an existing BigBank employee, joins the team after being interviewed by them. Because collaboration and teamwork are so important to agile teams, they need to take responsibility for determining who will join them. Their personnel department, or "human resources" department in many cases, can often narrow down the number of likely candidates for the team to consider. However, the team itself should make the final decision as to who becomes a member, not a manager who is external to the team. Doug is joining the team because Debbie is rotating onto another team. While the team has grown to become extremely efficient and intends to stay together long term, it is a good idea to rotate team members to keep them stimulated and to give them new learning opportunities. The experience bringing Doug onto the team was different enough for the personnel department to realize that agile required a different approach by them. This, in turn, motivated them to start learning about Disciplined Agile (DA)'s People Management process blade[2] and seek help from a qualified enterprise agile coach.

---

[1] See PMI.org/disciplined-agile/gci.

[2] See Ambler, S., & Lines, M. (2017). *An executive's guide to Disciplined Agile: Winning the race to business agility.* Disciplined Agile Consortium.

- **Parallel independent testing.** As Carlos suggested, toward the end of release one, the team works with BigBank's quality assurance (QA) team to have them begin testing new MAP builds in parallel to their development efforts. Negotiations to do this began in Iteration C1, but it takes almost two months for the QA team to make the right people, Thomas and Tatiana, available to do this testing. The challenge is finding people with sufficient skills to do this work. Many of the testers on the QA team are "old school," and need to work from a detailed requirements specification. That isn't going to work for agile parallel independent testing, because it is unlikely that there is going to be such a specification. Nor is it likely that the team is going to take on the overhead of doing so, simply to support this type of testing, and that was certainly the case with MAP. Instead, people with a more sophisticated testing skill set are needed, particularly around exploratory testing and end-to-end, cross-system integration testing. Carlos needs to work with Quincy, the QA manager, to educate him on how agile teams work and on the implications for his team. All of this takes time, and it isn't until the end of Iteration 4 that parallel independent testing begins. To report potential issues back to the team, Thomas and Tatiana capture them using a commercial test management tool, which everyone on the QA team uses. Ideally, they want to use the agile management tool they adopted, but Quincy wouldn't support that. In the end, Tara and Patricia need to review the reported issues. Tara captures them in the agile management tool so that the team can easily access them, and Patricia prioritizes them so that the issues are treated as another type of requirement. Quincy is surprised by how few defects were found by the independent testing team. Carlos explains that due to the greater focus of all team members on quality and testing each other's work, good agile teams don't let many defects "escape" the iterations. For agile teams, done means done—built and tested. Having said that, the independent testing effort was valuable because it found defects that the team didn't, in particular around cross-system integration bugs that likely would have escaped into production. This, in turn, enabled the team to identify weaknesses in their test strategy and then address them.
- **Vacations.** Danny takes two weeks off for vacation during Iteration C4, and Terry takes one week during Iteration C6. Members of agile teams, just like members of other teams, get to take vacation time. When they are gone, the team's capacity to perform work is correspondingly lower, something that is taken into account during iteration planning.
- **Sharing their learnings.** During the middle of Construction, BigBank began to roll out DAD to other delivery teams. Because the MAP team members were the most experienced DAD practitioners within BigBank, they were asked to share their knowledge with these other teams. At first, there was a push to move some people out of the team to help seed DAD knowledge into the other teams. This would have helped those teams, but would have decimated this one, and the decision had been made previously to see whether a long-lived (stable) team would work well within BigBank. Instead, the team decided to hold several "lunch-and-learn" sessions, where a team member would give a 10-minute talk about their experiences on a DAD team and then answer people's questions for the rest of the hour. BigBank had lunch brought in for each of these sessions to help support the effort. Carlos began to work with another DAD team, and two other Certified Disciplined Agile Coaches (CDACs)—Constance and Carter—were hired to help with BigBank's agile transformation efforts.

The Transition phase takes one week this release, down from two weeks the previous release. This is the result of the need for less training required for the support staff and better overall quality due to improved testing during Construction. Additionally, the team benefits from their investment in

automated deployment scripting and automated regression testing in the prior release. The training sessions were one hour each, compared with three hours each during the first release, allowing the team to schedule all of the training sessions on a single day. The team's approach to testing, in particular the adoption of TDD and parallel independent testing, dramatically reduces the need for hardening. However, there is still some hardening required, something that Carlos hopes to coach the team about in the next release.

Release 3

Release 3 is shortened to three months, with one week for Inception, six two-week Construction iterations, and one week for Transition. The following events of interest occurred during this release:

- **Training.** During Inception, the team attends a two-day training workshop in acceptance test-driven development (ATDD), also called behavior-driven development (BDD). After their experiences with unit-/design-level TDD during release two, team members feel that they should take the next step and start trying TDD at the requirements level.
- **Requirements envisioning.** Patricia runs a half-day requirements elicitation workshop with a group of stakeholders to flesh out new requirements. This occurs on the second day of the Inception phase, and the session is attended by all team members so that they can hear directly what the stakeholders wanted.
- **Improved quality.** A few minor issues, all severity three or four, are found during the warranty period following the second release. These issues are prioritized and added to the team's work item list. Victoria, the vice president of IT at BigBank, publicly commends the MAP team in one of her monthly newsletters for their high quality and stakeholder satisfaction ratings that they are consistently receiving.
- **Continuous deployment.** Although the team started to automate their deployment during the first release, it wasn't until now that it became truly continuous. By the end of Release 3, the team was able to set up deployment so that successful builds are automatically promoted from someone's workstation to their team integration environment, and from the team integration environment into their demo environment and a staging environment for the independent test team. Using feature toggles, the team starts to "turn on" smaller pieces of functionality on a more frequent basis so they can do more frequent deployments to production. By architecting these small changes as isolated microservices, the stakeholders feel more comfortable deploying these changes without large regression testing efforts.
- **Adoption of acceptance test-driven development (ATDD).** After their training, the team actively started to adopt ATDD tooling and techniques. Tara invested about half of her time during the first three iterations pairing with Barbara, Doug, and Danny, helping them to gain solid ATTD skills.
- **Organizational support for continuous integration (CI)/continuous deployment (CD).** BigBank's first step toward Disciplined DevOps is to form a small Center of Excellence (CoE)[3] comprised of three DevOps engineers to start. Their initial focus is on setting up a continuous development tool chain and helping the existing DAD teams to adopt their infrastructure. The MAP application team had these tools set up already, although the other teams newer to agile were still in the process of doing so.

[3] See PMI.org/disciplined-agile/people/centers-of-excellence.

- **Strategizing around DevOps.** The MAP team began working with the DevOps CoE to share ideas for how they could rearchitect the MAP application to support more effective development and operations of it. The team commits to a detailed discussion of this at the beginning of their next release.
- **Short Transition phase.** Transition is, once again, one calendar week. However, it actually took a bit less than three hours to run the regression test suites one more time and then run the deployment script. The rest of the week was the warranty period for that release (although no major errors were found), and the start of the Inception phase for release 4.

Releases 4+

The team continues improving the way that they work over time. The following key events occurred during release 4:

- **The team reworks MAP with DevOps capabilities.** The team decides to invest most of a week at the beginning of release 4 to learn about BigBank's DevOps strategy and what they need to do to fit into it. The team was interested in DevOps for two reasons: First, they wanted to delight their customers by providing better levels of service to them. Second, because DAD teams work in an enterprise-aware manner, it made sense to take advantage of BigBank's DevOps resources wherever possible. In release 4, their implementation of feature toggles expanded to address all functionality, not just the new features that were implemented in release 3. This required a bit of refactoring on the part of the team, reducing their ability to delivery new functionality during release 4 to make this investment. In release 5, the team added operational monitoring capabilities into the MAP application and ran an A/B test to help them identify which implementation of a new feature was preferable to their end users. Interestingly, they found this very easy to do, as the result of their existing support of feature toggles and with a framework provided by the DevOps CoE. In release 6, they added real-time chat into MAP to enable end-user feedback and support.
- **The release cadence was tightened.** To support Berhard's desire to shorten time to market, and to reduce overall risk on the MAP endeavor, the team actively tightened their release cadence over time: Release 4 was three months in length; releases 5 and 6 were both two months; and for release 7 and beyond, they settle on releasing every four weeks.
- **The life cycle evolves.** As the result of discussions during several retrospectives during release 5, the team decides to begin moving toward following DAD's Lean life cycle. This is viable because the team has relatively easy access to stakeholders and because the team has gelled so well. At the beginning of release 6, they choose to drop the concept of working in two-week iterations in favor of a continuous stream of development. Their first improvements are to hold weekly planning sessions and demos, and hold retrospectives whenever someone on the team decides to call one (because they have an issue to discuss). During release 8, the team replaces the weekly planning session in favor of replenishment sessions and pulls one work item at a time into their process (instead of a week's worth of work). The team is now capable of taking a new feature from the backlog and quickly building and deploying it to customers. This short cycle time reduces work in process and provides a quick return on investment.
- **Operating and supporting MAP.** With the team building, fixing, and deploying their own work, they have, in effect, adopted the DevOps practice and mentality of "you build it, you run it." Carlos points out that the team has successfully evolved into DAD's Continuous Delivery: Lean life cycle, which is the most advanced and effective delivery life cycle. This move dovetails well into BigBank's overall DevOps strategy.

- **The team stops sizing.** Part of the decision to settle on a one-month release cadence is to provide a consistent and predictable schedule to stakeholders. This enables the team to convince their stakeholders to allow them to adopt a #NoEstimates[4] strategy, where the team no longer takes on the overhead of sizing their work.

BigBank's Overall Disciplined Agile Transformation

The experiment with DAD on the MAP team is clearly a success. This was due to:

- BigBank's willingness to invest in training and coaching the team in Disciplined Agile techniques;
- The team's willingness to learn together to adopt the agile mindset;
- The team's willingness to try nonsolo work techniques such as pair programming and agile modeling, which enabled them to learn new skills faster; and
- Team members with good technical skills initially who were also willing to expand those skills as needed.

In parallel, five other teams are also deploying successfully with DAD, prompting BigBank to start adopting it across many development teams. This wider adoption started while the MAP team was in the middle of release 2. A year later, BigBank now has 15 teams following a DAD-based approach.

Although BigBank's initial DAD adoption was successful, it was clear to them that their inefficiencies went far beyond software development. They had been hearing about DevOps for a while now, and several of their new hires were eager to go in that direction, so BigBank decided to shift gears.

Having not learned their lessons from jumping on the Scrum bandwagon, they immediately decided to jump on the DevOps bandwagon. They invested in building a DevOps CoE, staffed by DevOps engineers who had experience with DevOps at smaller startup companies in the past. They started by promoting the classic team-based vision, as depicted in Figure 12.1, and by building out the tooling infrastructure to support CI/CD within delivery teams. Interestingly, the MAP team had

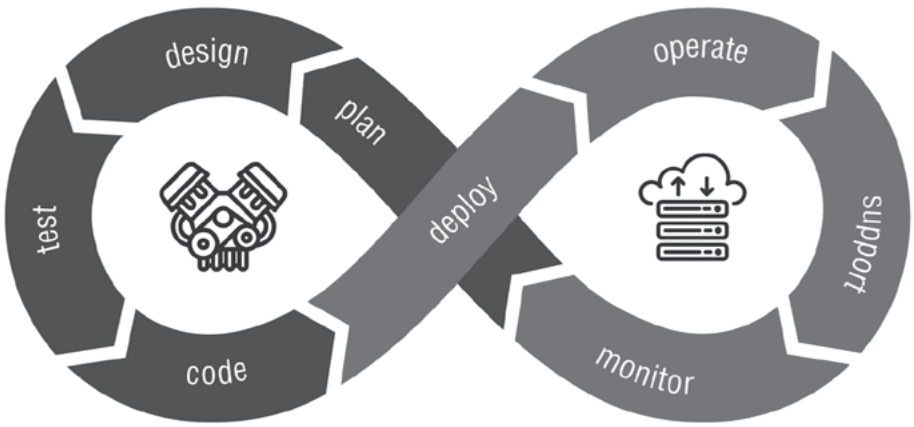


Figure 12.1 The DevOps loop.

[4] See NoEstimates.org/blog/.

been unknowingly evolving their approach in this direction already, so they, in effect, became one of the DevOps pilot teams for BigBank. DevOps requires a mindset that promotes collaborative, learning-centric ways of working that are supported by agile practices. On the development side, teams tend to adopt a continuous delivery (CD) approach and also take on many of the responsibilities that were the purview of operations and support organizations in the past.

Although the strategy depicted in Figure 12.1 is great for individual teams, it quickly failed in BigBank's multiteam, multiparadigm environment, where regulatory compliance and working with legacy systems are the norm. BigBank needed to ensure that security, data management, and fundamental business operations concerns were addressed by their DevOps strategy. Furthermore, the realities of having hundreds of teams working in parallel requires a bit of release management/coordination that goes beyond the team-based strategy. Similarly, their support/help desk strategy needed to address the fact that they had over 2,000 systems in production, tens of thousands of internal users, and millions of external users. Luckily, their Certified Disciplined Agile Coaches (CDACs) pointed them to Disciplined DevOps,[5] the workflow for which is depicted in Figure 12.2.

With the help of their DevOps Center of Excellence (CoE), BigBank was able to put the organizational, process, and tooling infrastructure in place to support a Disciplined DevOps strategy.

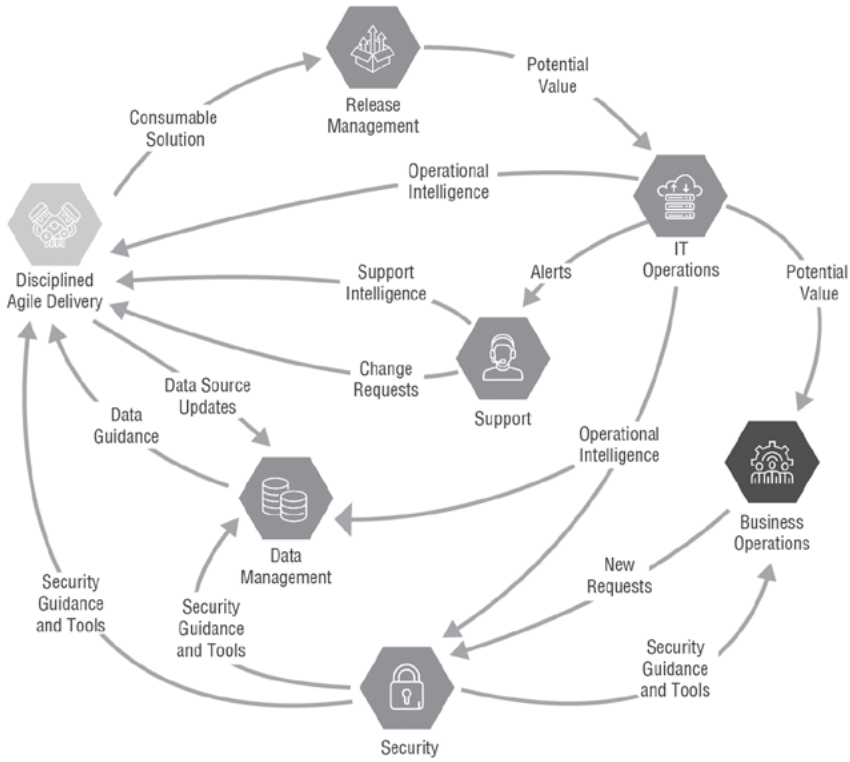


Figure 12.2 The full workflow of Disciplined DevOps.

[5] See PMI.org/disciplined-agile/process/disciplined-devops.

This took time and effort, but eventually they were able to build out an approach that reflected their enterprise context. Throughout this, the MAP team was an active partner of the DevOps CoE in that the CoE ran experiments with them to try out potential strategies before rolling them out to other teams. For example, the MAP team was the first one to try out the A/B testing framework developed by the CoE (during release 5) and the real-time chat framework (during release 6). In effect, working with the MAP team to work with the chat framework was the agile CoE's way to run an experiment to explore its viability in practice. The end result was that the CoE convinced BigBank to purchase an enterprise license for it. Behind the scenes, from the MAP team's point of view, the DevOps CoE coached the data management and security teams in how to work in an agile manner with the MAP team. BigBank's Disciplined DevOps strategy proved to be a good start, but was insufficient in the long run. They quickly realized that the rest of their IT strategy needed to evolve to support business agility throughout their entire organization. This required them to adopt a Disciplined Agile IT (DAIT) strategy to support their overall Disciplined Agile Enterprise (DAE). These two aspects of the Disciplined Agile (DA) tool kit are overviewed in the appendix of this book.

Chapter 13

Closing Thoughts

We'd like to leave you with a few ideas around why you should consider adopting the Disciplined Agile (DA) tool kit, where you can learn more about DA, and how you can get some help becoming more disciplined in your approach to agile solution delivery.

Why DA?

You should consider adopting a Disciplined Agile approach when:

- You want a flexible and pragmatic agile tool kit, rather than a purist agile method.
- You are successfully using Scrum or Scrum/XP and want to take it to the next level.
- You are using agile, but have not been getting the results that you expected.
- You are out of compliance with your PMO and want to incorporate some lightweight agile governance.
- You are using Scrum, but are unsure of how to scale up.
- You are using Scrum and are unsure of how to address fundamental activities such as architecture, testing, and analysis.
- You have looked at SAFe, but it appears to be too expensive and risky for your organization.
- You have looked at either SAFe or LeSS, but realize that DA's strategy of starting with the organization you have and evolving from there is less risky and more likely to succeed in your organization.
- Your organization has adopted SAFe, but run aground because you didn't have a solid foundation in place for your agile delivery teams.
- You need to support several approaches to agile/lean development within your organization.
- You need to understand how to effectively blend agile/lean initiatives with your teams that use a traditional approach.

How to Learn More

- As DA continues to evolve, the latest material can be found at PMI.org/disciplined-agile.
- The certification program and upcoming DA training can be found at the Disciplined Agile Hub, PMI.org/what-is-disciplined-agile.
- The PMI book *Choose Your WoW! A Disciplined Agile Delivery Handbook for Optimizing Your Way of Working* describes DAD in detail, showing how to take a guided continuous improvement (GCI) approach to improving your team's way of working (WoW).
- The PMI book *An Executive's Guide to Disciplined Agile: Winning the Race to Business Agility* provides an excellent overview of the tool kit.

Appendix

The Disciplined Agile Tool Kit

The Disciplined Agile (DA) process improvement tool kit provides lightweight guidance to help organizations streamline their information technology (IT) and business processes in a context-sensitive manner. It does this by showing how the various activities, such as solution delivery, operations, enterprise architecture, portfolio management, finance, security, legal, and many others work together. The tool kit also describes what these activities should address, provides a range of options for doing so, and describes the trade-offs associated with each option. In effect, DA provides the process foundation for business agility. As you can see in Figure A.1, there are four levels to the DA tool kit:

1. **Foundation.** The foundation layer provides the conceptual underpinnings of the DA tool kit.
2. **Disciplined DevOps.** DevOps is the streamlining of solution development and operations, and Disciplined DevOps is an enterprise-class approach to DevOps. This layer includes Disciplined Agile Delivery (DAD), the focus of this book, plus other enterprise aspects of DevOps. The workflow is overviewed in Figure A.2.
3. **Value Stream.** The value stream layer is based on Al Shalloway's FLEX. It's not enough to be innovative in ideas if these ideas can't be realized in the marketplace or in the company. FLEX is the glue that ties an organization's strategies in that it visualizes what an effective value stream looks like, enabling you to make decisions for improving each part of the organization within the context of the whole. An overview of the workflow is shown in Figure A.3.
4. **Disciplined Agile Enterprise (DAE).** The DAE layer focuses on the rest of the enterprise activities that support your organization's value streams.

From Scaling to Enterprise Agile

What does it mean to scale agile? The answer to this question depends on whom you ask. For example, some people will tell you that scaling agile means applying agile strategies to a large software development team or to a geographically distributed software development team. To others, scaling agile means applying agile strategies across a lot of software development teams, while to others, scaling agile means you apply agile strategies to your organization as a whole. As a result, the DA tool kit distinguishes between two types of "agility at scale":

1. **Tactical agility at scale (to deal with context).** This is the application of agile and lean strategies on individual Disciplined Agile Delivery (DAD) teams. This includes the ability to apply agile on teams of all sizes, on teams that are geographically distributed, on teams facing regulatory compliance, on teams addressing a complex domain (problem space), on teams applying complex technologies, on teams where outsourcing may be involved,

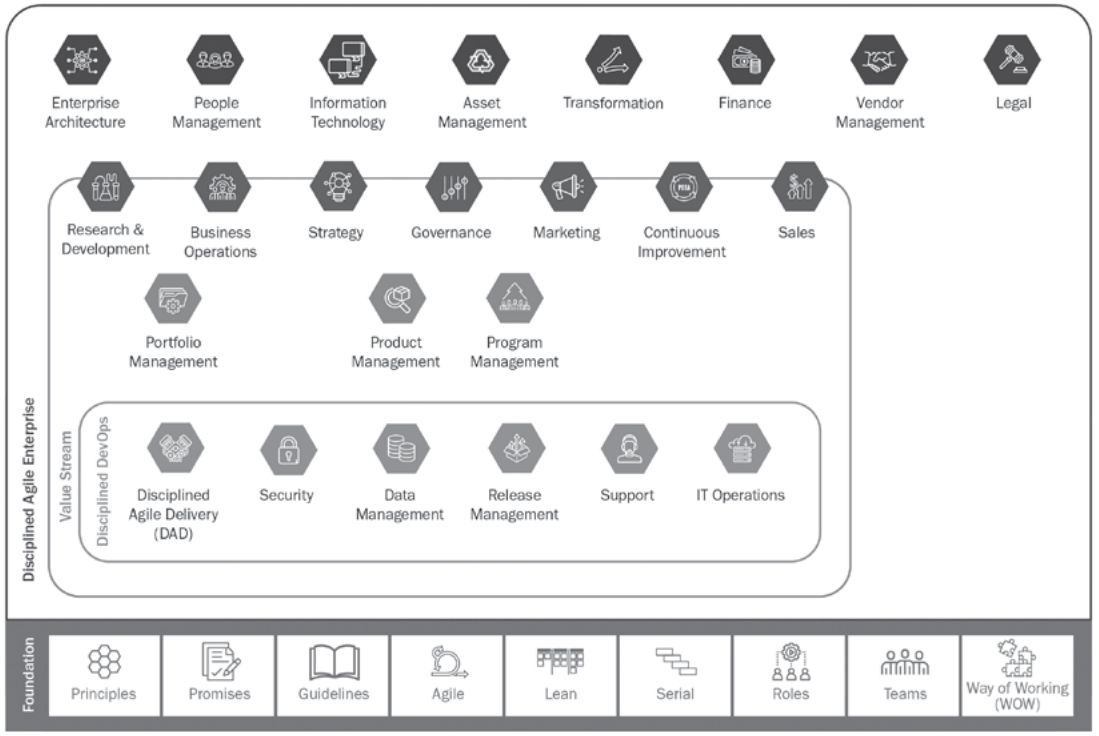


Figure A.1 The scope of the DA tool kit.

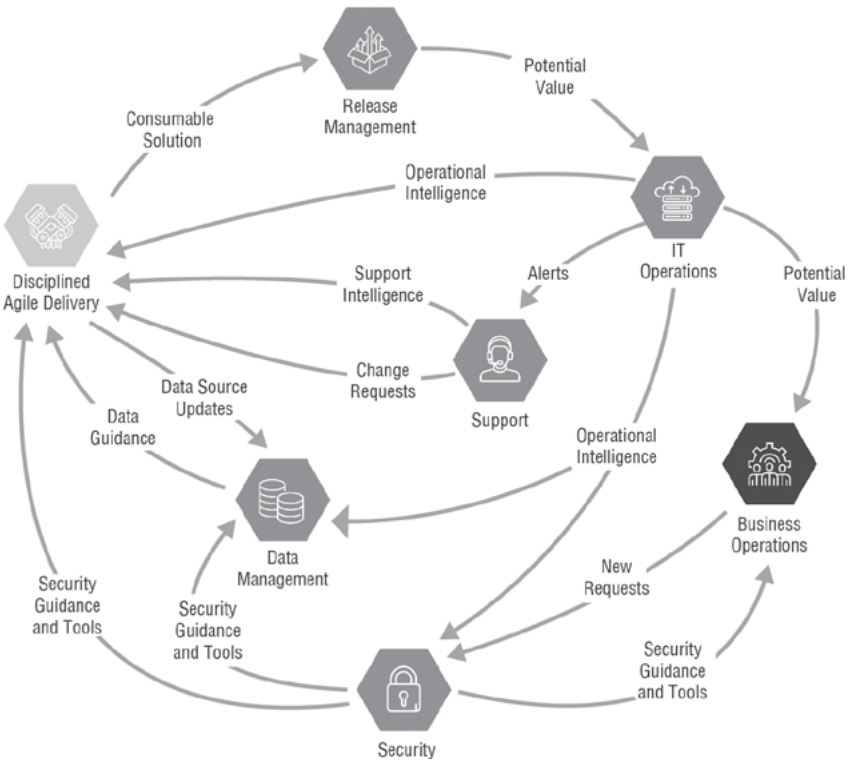


Figure A.2 The workflow of Disciplined DevOps.

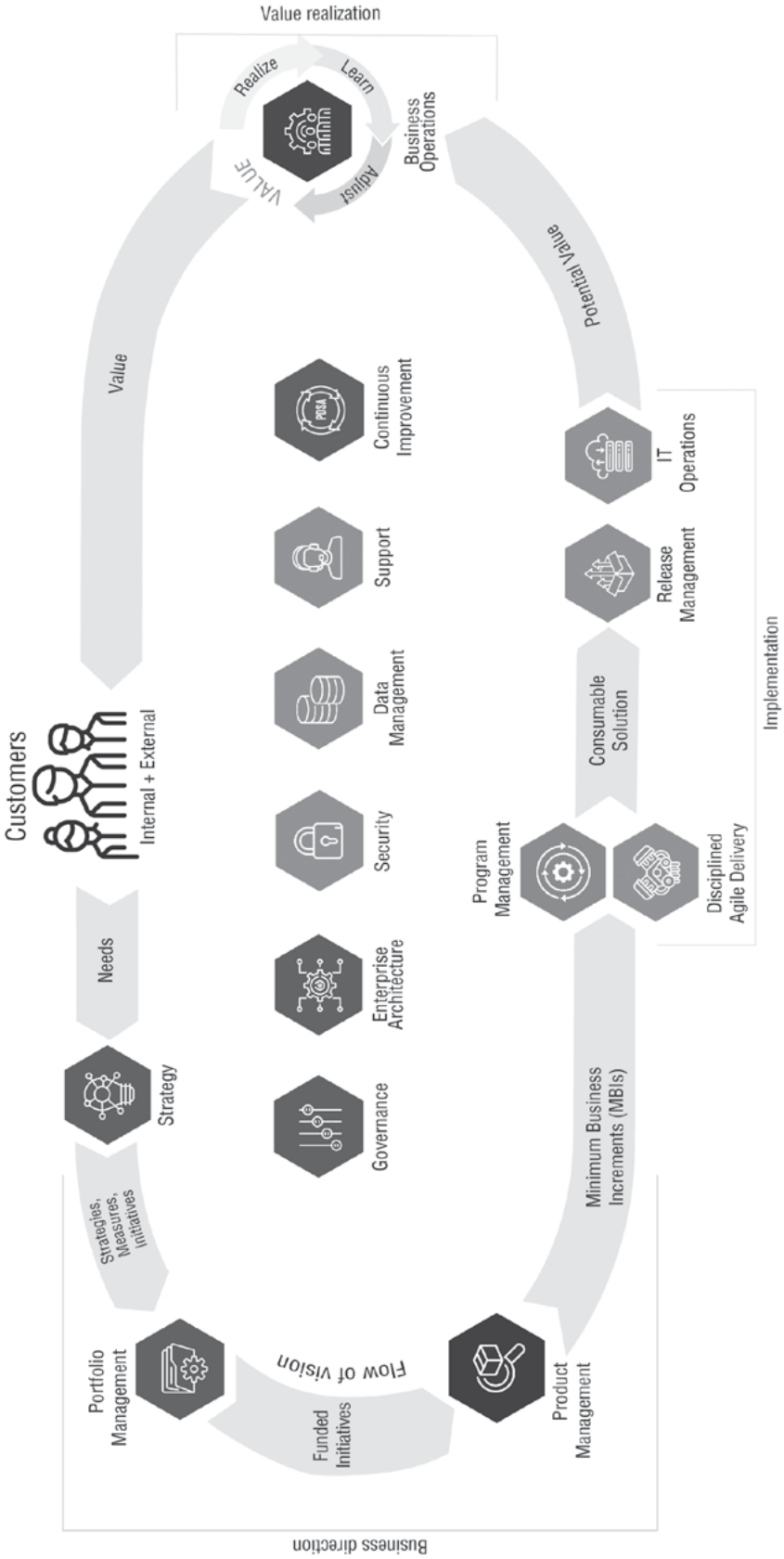


Figure A.3 The DA FLEX life cycle for value streams.

and combinations thereof. An important implication of this is that because you are likely to have delivery teams facing different situations, these teams will be following different tailorings of DAD—context counts.

2. **Strategic agility at scale (to deal with organizational breadth).** This is the broad application of agile and lean strategies across your entire organization. From an IT point of view, this includes Disciplined DevOps and Disciplined Agile IT (DAIT), in general. From an enterprise point of view, this includes all divisions and teams within your organization, not just your IT department, what we call the Disciplined Agile Enterprise.

References

Agile Data. (n.d.). AgileData.org

Agile Modeling (n.d.). AgileModeling.com

Ambler. S., & Lines, M. (2017). *An executive's guide to Disciplined Agile: Winning the race to business agility.* Disciplined Agile Consortium.

Ambler, S., & Lines, M. (2020). *Choose your WoW! A Disciplined Agile Delivery handbook for optimizing your way of working (WoW).* Project Management Institute.

Ambler, S., & Sadalage, P. (2006). *Refactoring databases.* Addison Wesley.

Ambysoft. (2016). *2016 agility at scale survey results*. Ambysoft.com/surveys/agileAtScale2016.html

Disciplined Agile Consortium. (n.d.). DisciplinedAgileConsortium.org

Disciplined Agile. (n.d.). *Guided continuous improvement.* PMI.org/disciplined-agile/gci.

Disciplined Agile. (n.d.). *Centers of excellence.* PMI.org/disciplined-agile/people/centers-of-excellence.

Disciplined Agile. (n.d.). *Disciplined DevOps.* PMI.org/disciplined-agile/process/disciplined-devops.

Feathers, M. (2004). *Working effectively with legacy code.* Prentice Hall.

NoEstimates. (n.d.). NoEstimates.org/blog/

Patton, J. (2014). *User story mapping*. O'Reilly Media.

Ries, E. (2011). *The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses*. Crown Business.

Index

#
#NoEstimates, 72

A
acceptance TDD, 70
agile life cycle, 10
Agile Modeling, 5
agile release train, 16
agility at scale, 22, 77
agnostic, 6
architecture
  envisioning, 30
  handbook, 32
  technology roadmap, 49
  walking skeleton, 48
architecture owner, 6
automated deployment, 58

B
BDD, 70
burnup chart, 54

C
certification, 2, 75
choice is good, 23
  life cycles, 6
CMMI, 22
coaching, 30, 72
colocate, 36
commitments, 42
consumable solutions, 10
context counts, 18, 23
continuous delivery, 73
continuous delivery agile life cycle, 10
continuous delivery lean life cycle, 13
continuous delivery life cycle, 71
continuous development, 70
continuous improvement, 68
continuous integration, 3, 13, 47, 57
continuous release stream, 16
coordination meeting, 39, 48

D
dashboard, 58
data modeling, 31, 43, 56
defect management, 61, 65, 69
defect trend chart, 61
definition of done, 41
demo, 44, 49, 62
deployment planning, 58, 61
deployment testing, 65
development environment, 37
development intelligence, 58
DevOps
  automated deployment, 58
  tooling, 70
  you run it, 71
Disciplined Agile (DA) tool kit
  DAE, 74
  DAIT, 74
  DevOps, 73
  scope, 77
Disciplined Agile (DA) tool kit
  DAD, 1
Disciplined Agile Delivery, 5
Disciplined Agile IT, 74
documentation, 55
  architecture handbook, 32
  continuous, 56, 58, 65
  deliverable, 65
dropping scope, 43

E
enterprise architecture, 31
exploratory life cycle, 13
Extreme Programming, 5

F
fail fast, 22
feature toggles, 31

G
Gantt chart, 35
GCI, 68
goal diagram
  explore scope, 18
  plan the release, 35
  produce a potentially consumable solution, 50
goal driven, 18
  Inception phase, 29

governance, 4, 6, 8, 22, 75
guided continuous improvement, 68

H
hardening, 43, 49, 54, 59, 70
  hardening sprint anti-pattern, 63
hybrid, 6

I
Inception phase, 29
ISO, 22
IT governance
  milestones, 8
iteration burndown, 42
iteration planning, 39, 47
iteration review, 43
ITIL, 22

J
just in time design, 47

K
Kanban, 5

L
lean life cycle, 10, 71
Lean Software Development, 5
Lean Startup, 13
learn fast, 22
LeSS, 16
life cycle
  agile, 10
  and process improvement, 17
  continuous delivery agile, 10
  continuous delivery lean, 13
  exploratory, 13
  lean, 10
  risk value, 22
long-lived teams. *See* stable teams
look-ahead modeling, 47

M
measured improvement, 68
metrics
  automated dashboard, 58
  defect trend chart, 61
  release burnup chart, 54, 58
  velocity, 35, 44, 51
microservices, 31, 70
milestone
  delighted stakeholders, 35
  proven architecture, 49
  stakeholder vision, 38
  sufficient functionality, 55, 63
milestones, 8
mindset
  learn fast, 22
modeling
  data, 43
  initial architecture, 29
  initial requirements, 30
  just in time, 47
  look-ahead analysis, 45, 48, 53
  look-ahead design, 45
  up-front, 56
  usage, 30
MVP, 13

N
Nexus, 16
nonfunctional requirements, 32
nonsolo work, 53, 72

O
optimize flow, 17

P
pairing, 53
parallel independent testing, 62, 69
phases, 8
planning
  date-driven, 35
  deployment planning, 58, 61
  iteration planning, 39, 47
  release planning, 33, 58, 66
  scheduling, 35
  scope-driven, 35
planning poker, 33
PMO, 38
pragmatic agile, 1
principle
  choice is good, 6, 23
  context counts, 18
  optimize flow, 17
prioritization, 32, 66

process
  goal driven, 18
process improvement
  and life cycles, 17
product teams. *See* stable teams
proof of concept, 33

Q
quality assurance, 69

R
refactoring
  legacy code, 31
regulatory compliance
  separation of roles, 66
relative-mass sizing, 33, 54
release burnup chart, 54
release planning, 33, 58, 66
release window, 58
remote work, 68
requirements
  ATDD/BDD, 70
  envisioning, 30, 66, 70
retrospective, 44, 58, 62
  measured improvement, 68
risk management, 32
  delighted stakeholders, 35
  deployment risk, 31
  mitigation of risk early, 33
  proven architecture, 49
  release cadence, 71
  stakeholder vision, 38
  sufficient functionality, 55, 63
risk value life cycle, 22
roles, 6

S
SAFe, 6, 16
scaling, 77
  factors, 22
  strategic, 22, 80
  tactical, 22, 77
scope creep, 53
screen sketches, 48
Scrum, 18
spike, 33
stabilization, 43
stable teams, 13, 17, 69
stakeholders, 35
standup meeting. *See* coordination meeting
story map, 30
strategic scaling, 22, 80
succeed early, 22

T
tactical scaling, 22, 77
task board, 39
team kickoff, 29
team of teams, 13
team room, 36, 68
technology roadmap, 31, 49
test often, 57
test-after programming, 51
test-driven development, 67
test-first programming, 51
testing
  deployment testing, 65
  final testing, 65
  parallel independent testing, 62
  test strategy, 30
testless programming, 50
time to market, 71
training, 20, 72
  end users, 65
  stakeholders in DAD, 57
  support staff, 58, 61, 65
Transition phase, 65
transparency, 43

U
Unified Process, 5, 8
up-front planning, 3, 33
user stories, 39

V
vacations, 69
velocity, 35, 49
  and defects, 61
vision statement, 37

W
walking skeleton, 48
warranty period, 67
WaterScrumFall, 3
working from home, 68

About the Authors

Scott W. Ambler is the vice president and chief scientist for Disciplined Agile at Project Management Institute where he leads the evolution of the DA tool kit. Scott is the cocreator, along with Mark Lines, of the Disciplined Agile (DA) tool kit and founder of the *Agile Modeling (AM)*, *Agile Data (AD)*, and *Enterprise Unified Process (EUP)* methodologies. He is the co-author of several books, including *Disciplined Agile Delivery*, *Refactoring Databases*, *Agile Modeling*, *Agile Database Techniques*, *The Object Primer – Third Edition*, and many others. Scott is a frequent keynote speaker at conferences, he blogs at ProjectManagement.com, and you can follow him on Twitter via @scottwambler.

Mark Lines is the vice president for Disciplined Agile at Project Management Institute and a Disciplined Agile Fellow. He is the cocreator of the DA tool kit and is a co-author with Scott Ambler of several books on Disciplined Agile. Mark is a frequent keynote speaker at conferences and you can follow him on Twitter via @mark_lines.

Introduction to Disciplined Agile Delivery—Second Edition

How does a small development team in an established enterprise apply Disciplined Agile (DA) strategies to successfully improve their agility and provide real value to their stakeholders? Find out how in *Introduction to Disciplined Agile Delivery*—Second Edition. This guide to Disciplined Agile Delivery (DAD) provides the foundation from which to scale agile and lean software development strategies, enabling teams to succeed in the unique situations they face—remembering that context counts.

Starting with a Scrum-based approach, we'll show you how the team learns from their experiences and evolves into a lean life cycle, then to a continuous delivery life cycle within a DevOps environment. We begin with an overview of DAD and work through a case study that describes a team's learnings through several releases of a mission-critical solution.

You'll find out how DAD:

- Gives you the flexibility to use various approaches while covering gaps not addressed by mainstream agile practices.
- Describes proven strategies that show how programming, design, testing, architecture, analysis, deployment, and many more aspects of solution delivery fit together in a streamlined whole.
- Shows you how to turn the agile software development dial all the way up!

Mark Lines and Scott W. Ambler

Mark Lines and Scott W. Ambler are cocreators of PMI Disciplined Agile and authors of several agile books. They have decades of experience implementing agile and lean at organizations around the world and are both sought-after keynote speakers.

ISBN: 978-1-62825-654-3

9 781628 256543 51495

PMI Disciplined Agile